|  |
| --- |
| Anglia Ruskin University |
| Group Design Project |
| Group 4 |

|  |
| --- |
|  |

Contents

[Interview 4](#_Toc260852785)

[Hardware and Networking Solutions 13](#_Toc260852786)

[Hardware Required 13](#_Toc260852787)

[Network Topology 13](#_Toc260852788)

[Data Dictionary 15](#_Toc260852789)

[Data Flows 15](#_Toc260852790)

[Data Stores 16](#_Toc260852791)

[Forms Data Dictionary 16](#_Toc260852792)

[Data Flow Diagram 18](#_Toc260852793)

[Level 0 18](#_Toc260852794)

[Level 1 19](#_Toc260852795)

[Chen Diagram V1.0 20](#_Toc260852796)

[Database Schema 20](#_Toc260852797)

[Initial Entity Relationship Diagram 21](#_Toc260852798)

[Chen Diagram 22](#_Toc260852799)

[Chen Diagram V2.0 23](#_Toc260852800)

[Database Schema 23](#_Toc260852801)

[Initial Entity Relationship Model 24](#_Toc260852802)

[Chen diagram 25](#_Toc260852803)

[Chen Diagram V3.0 26](#_Toc260852804)

[Database Schema 26](#_Toc260852805)

[Initial Entity Relationship Model 27](#_Toc260852806)

[Chen diagram 28](#_Toc260852807)

[Software Functions 29](#_Toc260852808)

[Website Functions 37](#_Toc260852809)

[Coding 39](#_Toc260852810)

[Testing Methods 82](#_Toc260852811)

[Black Box Testing 82](#_Toc260852812)

[White Box Testing 84](#_Toc260852813)

[Login 84](#_Toc260852814)

[Menu 84](#_Toc260852815)

[Staff Functions 87](#_Toc260852816)

[Managements 92](#_Toc260852817)

[Login Option 97](#_Toc260852818)

[Usability Testing 98](#_Toc260852819)

[1. Visibility of System Status 98](#_Toc260852820)

[2. Match Between System and the Real World 101](#_Toc260852821)

[3. User Control and Freedom 103](#_Toc260852822)

[4. Consistency and Standards 105](#_Toc260852823)

[5. Help Users Recognize, Diagnose, and Recover From Errors 109](#_Toc260852824)

[6. Error Prevention 111](#_Toc260852825)

[7. Recognition Rather Than Recall 112](#_Toc260852826)

[8. Fexibility and Minimalist Design 115](#_Toc260852827)

[9. Aesthetic and Minimalist Design 117](#_Toc260852828)

[10. Help and Documentation 118](#_Toc260852829)

[11. Skills 120](#_Toc260852830)

[12. Pleasurable and Respectful Interaction with the User 122](#_Toc260852831)

[13. Privacy 124](#_Toc260852832)

[Website Testing 125](#_Toc260852833)

[Website Test Plan - Firefox 126](#_Toc260852834)

[Website Test Plan - Chrome 127](#_Toc260852835)

[Website Test Plan - Internet Explorer 128](#_Toc260852836)

[Website Test Log - Firefox 129](#_Toc260852837)

[Website Test Log – Chrome 130](#_Toc260852838)

[Website Test Log - Internet Explorer 131](#_Toc260852839)

[Overview 132](#_Toc260852840)

# Interview

**1. Do they use any software solution currently?**

No. But they keep the member details in a crude access 2003 tale.

Everything else is booked manually.

**2. What kind of information handled/received by staff interactions during work?:**

* Induction booking in manual folder
* He also said member data input happens always after actual induction, because it initiates the card access process
* Class booking requests
* Trainer booking requests
* Trainer details
* Resource details (like minivan renting)

**3. What data do you record about members?**

- FirstName   
- LastName   
- SID   
- MembershipType (Student, Staff, Gym Member, SportsFederation, Gym + SportsFederation, Gym Alumni, Gym Shorterm, Gym SummerSchool, Community Monthly, Community PAYG)   
- Gender   
- Expire Date   
- Renewal (YES, NO)   
- Address   
- Email   
- Phone Number   
- Emergency Number   
- Nationality   
- Receipt Number   
- Money Taken (initials of the person who took the money)   
- Club (if applicable)

**4. What data do you record about trainers?**

-name  
-Date of birth  
-qualifications (what classes can they do)  
-inductioning (yes/no)  
-personal training (yes/no)

**5. What are the gym business hours (when is it open for member access)**

8am-10pm

**6. What are the shift and lunch times?**

- Shifts work from:   
- Early shift: 7:30 till 2:30/3:30   
- Late shift: 1pm till 8pm   
Lunch time: 1pm till 1:30pm



**Gym Floor Plan 1.1**

Scenario of Current System

Classes run up to 3 times a week

& Machines

10 Classes

Inductions

2 Managers

Customer Payments

200 Members

15 Fitness Instructors

2 Receptionists

Current System

Trees leisure centre & gym is independently run and has no current electronic system for recording time tables, memberships or payment. There is an access database simple containing membership contact details but everything else is recorded by hand and kept on paper. The staff books the inductions manually in an appointment book. There are 2 managers, 2 receptionists and 15 other members of staff, all the contact details and any other information is all stored on paper in the manager office. The gym has fitness machine which can be used by all of gym members, however a member can also book a personal trainer for 2 hour blocks throughout the day. The leisure centre also offers 10 other classes including yoga, spinning class and aerobics, a lot of the classes take place up to 3 times a week to meet the demand of the sym members. Each class can only have a certain number of members; each member can book themselves on a class however a certain number of spaces are always left empty to allow for walk-in’s. If a member wishes to attend any of the classes or book a personal trainer they either phone the reception desk to arrange times or physically visit the gym. At present the record of payment is all on paper and the only information recorded is the initials of receiving staff.

The trainers are employed specifically for the time constraints of the classes, therefore the system needs not to care about individual trainer availability as they sort out between each, if there is more than one trainer for a class.

Currently there are no access rights at all, anyone can start up the access database and modify any data or write anything into the manual records. We also propose at least three user access level: manager, staff and customer (for booking class over internet).

**Rich Picture**

Database

Servers

Networking

Online system

Class Booking

Timetabling

Rotas

Back office systems

Log in/log out

Terminals

Implementation

Maintenance

Training

Machinery/equipment

Facilities

Classes available

Class users

Casual gym users

Regular gym users

Software

Existing Features

Market

Cost

Hardware

**Gym Management System**

Use Case

Amend Class List

Amend Staff List

Manager

Amend Room List

Amend Booking

Amend Member List

Issue New Card

Renew Membership

Receptionist/

Fitness Instructor

Send Feedback

Amend Details

Member

Book Class

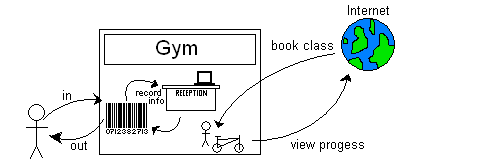
View Progress

Proposed System

The proposed system will be able to hold the staff rotas, which will be set and edited by the manager and viewed by the members of staff. The manager will also be able to add new classes to the system for example if the gym decided to start teaching a spin class then the information about the class would need to be added to the system so that members could be booked on that particular class. The program also needs to be able to book members on class induction and/or personal training session. There needs to be a facility that can record information about new members, and then record the member’s progress throughout their membership. For example it would include classes they have attended and target weight.

There needs to be access rights implemented within the gym to make sure that the system is secure and that all information is kept safe. Different security levels can be controlled by giving certain user names access to only certain areas of the system. For example fitness instructors can only view staff rotas, enrol members on a class, view class information, view member information and alter member information, where as the manager can do all of this and have the added privileges of editing staff information, staff rotas, class information and adding brand new classes to the system.

This will be implemented on to two computers situated at the front desk, for use by the receptionist and the fitness instructors. There will also be a computer in the manager’s office so that they can do work in private.



Gantt chart

We have constructed two gantt charts, one of these gantt charts has been constructed One of these gantt charts was constructed in Microsoft project that particular gantt chart shows just the stages of the entire process from the start of the requirement analysis until the project has been completed and implemented. However it does to specify tasks it is just a top level plan.

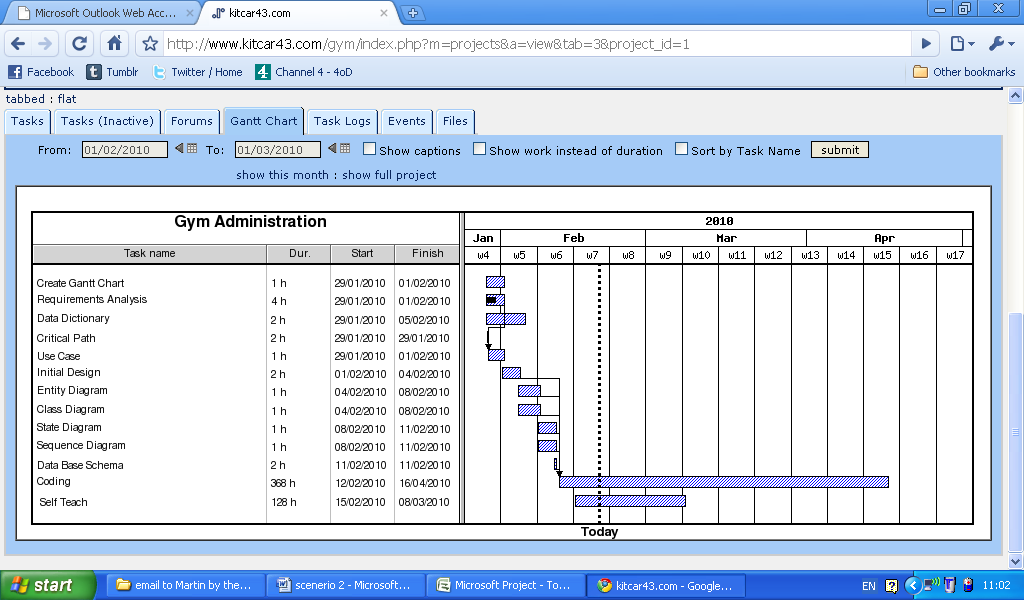
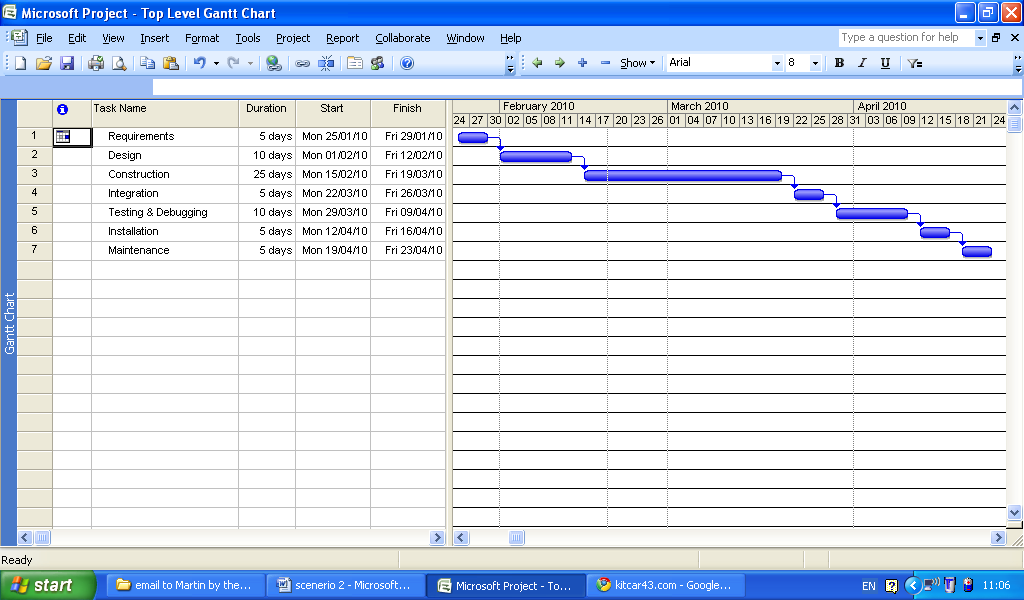
The other however was constructed using a piece of project management software called ‘dot project’, we have chosen this software because it is accessed online and can be accessed and edited by any member of the group. Each task is given a start and end date, it is also required that a member the group to be assigned to completing the task, more than one member can be assigned to a task. This means that if a new task is decided then any member can log on and assign themselves to that task. It keeps the plan up to date and allows any member of the group to check the time or place of a meeting. Dot Project also has the capability to upload documents and share them with each member of the group, a member can edit that document and upload a new version. This means that we can study any changes to make sure we don’t lose any important information.

Microsoft Project

Microsoft Project is a project management software program it is designed to assist project managers. The program allows you to track project progress, budgets, resources, developments and analyzing workloads.

The program also allows you to create critical path schedules from the information that is entered. There is also user options so that different classes of users can be defined to control different access levels.

Resources can easily be assigned to different tasks to make sure that deadlines are met & all users are aware of what needs to be done. Finances can easily be controlled which can improve cost estimates.

 ****

Critical Path

15

5

0

5 Days

5 Days

10 Days

5 Days

25 Days

10 Days

5 Days

Design

8

Requirements

Construction

Integration

Testing &

Debugging

Installation

Maintenance

7

6

5

4

3

2

1

The critical path method or critical path analysis is a mathematically based algorithm for scheduling a set of project activities. It is an important tool for effective project management.

The technique for CPM includes the following:

1. A list of all activities required to complete the project
2. The time that each activity will take to complete.

|  |  |  |
| --- | --- | --- |
| Task | Length | Dependencies |
| Requirements | 5 Days |  |
| Design | 10 Days | Requirements |
| Construction | 25 Days | Design |
| Integration | 5 Days | Construction |
| Testing & Debugging | 10 Days | Construction |
| Installation | 5 Days | Testing & Debugging |
| Maintenance | 5 Days | Installation |

1. The dependencies between the activities.

Using these values CPM calculates the longest path of the planned tasks from start to finish. It also includes the start and finish times for each task what extending the length of the project. The process determines which tasks are critical and which tasks can be delayed.

The result of the critical path analysis allows managers to practice tasks to make sure that the completion of the task is time effective and target met.

Maintenance

Testing &

Debugging

Integration

Construction

Requirements

Design

Installation

1

2

3

47

57

6

7

8

# Hardware and Networking Solutions

As with any computer system, the existing hardware must be evaluated before any implementation of software or future hardware solutions. As the current system was so simple and paper based, the client really had no reason to have more than two computers. One for the upkeep of the database, and one for simple word processing, for posters and the creation of timetables etc.

With the introduction of the new system, with its desktop application and its web based application, the old hardware would be unable to keep up with the demands of the new software system. We have therefore devised the following solution for the client.

## Hardware Required

To enable the client to be entirely computer based, we have concluded that the following hardware options are to be sourced.

Server – Dell Poweredge T110 – An affordable and robust server, ideal for small businesses. [[1]](#footnote-1)

Desktop Computers – Dell Vostro Desktop Computer – perfect for a small business like this.[[2]](#footnote-2)

Router – Cisco 800 Series – Ideal for small businesses.[[3]](#footnote-3)

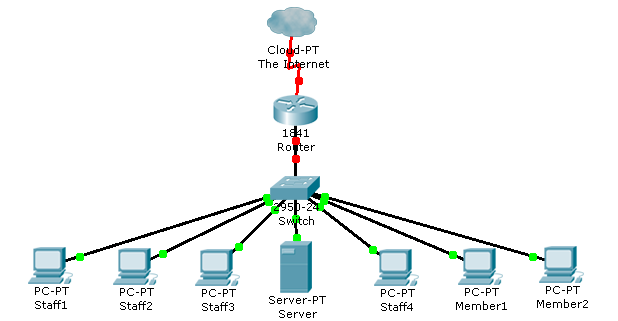
Switch – Cisco Catalyst Express 500 Series – once again, ideal for a small business.[[4]](#footnote-4)

All of the above products have been chosen because they are ideally suited for a small business such as this. The Dell server is optimised for use with less than 25 other machines, and seeing this system would only require 4 (6 if the existing machines are to be replaced) currently, this server seems a better option than some of the higher costing ones. The Dell Vostro Desktop, like the server, is ideal for a small business. It is affordable, and the one chosen is the slim tower design, which means it will reduce the amount of space needed to store it.

The router and switch, the fundamental items behind collaborating the network for the client. Both of these have been selected from the broad range of products by Cisco, simply because of the manufacturer’s reputation within the networking community.[[5]](#footnote-5) Whilst the comments are not corresponding to the chosen hardware in this scenario, they apply to the company as a whole. However the chosen products are perhaps more suited towards the small office than small gym, but we feel that they can cover the requirements substantially.

## Network Topology

One other fundamental thing within this system is how the computers, server, router and switch will be connected. Below is the proposed network topology for the client.

[[6]](#footnote-6)

Member1 and Member2 are currently the old machines, but in the future these could be used by members to book their own classes and change their own details.

The IP addressing would consist of a single subnet, with an ideal network address of 192.168.1.0/24. This then could be adapted if any future expansion was considered, simply by adding more subnets.

# Data Dictionary

## Data Flows

|  |  |  |
| --- | --- | --- |
| **Ref** | Data | *Description* |
|  | address | = houseNumber + postCode |
|  | classBookingInfo | = { trainerName + className + roomNumber + date + time } |
|  | classBookingRequest | = trainerName + className + roomNumber + classStartDate + classStartTime + classLength |
|  | classStartDate | = date |
|  | date | = day + month + year |
|  | day | \*number between 1-31\* |
|  | gender | = [ Male | Female ] |
|  | name | = (nameTitle) + { [ forename | initial ] } + surname |
|  | nameTitle | = [“Mrs” | “Ms” | “Mr” ] |
|  | sessionType | = [ Induction | Personal Training ] |
|  | trainerBookingInfo | = { trainerName + sessionType + roomNumber + sessionStartDate + sessionStartTime + sessionLength } |
|  | trainerBookingRequest | = trainerName + sessionType + roomNumber + sessionStartDate + sessionStartTime + sessionLength |
|  | trainerName | = name |
|  | year | \*the current year\* |

## Data Stores

|  |  |  |
| --- | --- | --- |
| **Ref** | Data | *Description* |
| **M1** | memberRecord | = { memberDetails } |
| **M2** | TrainerAppBook | = trainerBookingInfo |
| **M3** | ClassTimeTable | = classBookingInfo |
| **M4** | TrainerRecords | = { trainerDetails } |

## Forms Data Dictionary

|  |  |  |
| --- | --- | --- |
| **Ref** | Data | *Description* |
| **Member Details Form** | | |
|  | expireDate | = date |
|  | memberDetails | = memberName + SID + membershipType + gender +   expireDate + renewal + address + email + phoneNumber +   emergencyNumber + nationality + receiptNumber +   moneyTaken + ( club ) |
|  | memberDetailsForm | = memberDetails |
|  | memberDetailsFormBlank | = memberDetailsForm  \*Blank form\* |
|  | memberDetailsFormFilled | = memberDetailsForm  \*Filled with valid data \* |
|  | membershipType | = [ Student | Staff | Gym Member | SportsFederation | Gym And SportsFederation | Gym Alumni | Gym Shortterm | Gym SummerSchool | Community Monthly | Community PAYG ] |
|  | moneyTaken | = { initials }  \* initials of the person who took the money\* |
|  | Renewal | = [ yes | no ] |
| **Trainer Details Form** | | |
|  | dateOfBirth | = date |
|  | inductioning | = [ yes | no ] |
|  | personalTraining | = [ yes | no ] |
|  | trainerDetails | = trainerName + dateOfBirth + qualifications + inductioning + personalTraining |
|  | trainerDetailsForm | = trainerDetails |
|  | trainerDetailsFormBlank | = trainerDetailsForm  \*Blank form\* |
|  | trainerDetailsFormFilled | = trainerDetailsForm  \*Filled with valid data \* |

# Data Flow Diagram

## Level 0



## Level 1



# Chen Diagram V1.0

## Database Schema

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | STAFF | | **id\_staff** | | **id\_user \*** | | name | | address | | position | | NInumber | | contractType | | phoneNr |  |  | | --- | | EQUIPMENT\_BOOKINGS | | **id\_mbr\_eq\_booking** | | **id\_member \*** | | **id\_equipment \*** | | date\_start | | date\_due | | |  | | --- | | MEMBERS | | **id\_member** | | **id\_user \*** | | name | | address | | email | | type | | payment\_method | | phoneNr |  |  | | --- | | ROOMS | | **id\_room** | | name | | description | | size | | |  | | --- | | CLASSES | | **id\_class** | | name | | description | | type |  |  | | --- | | CLASS\_BOOKINGS | | **id\_class\_booking** | | **id\_class\_instance \*** | | **id\_member \*** | | booking\_date |  |  | | --- | | CLASS\_INSTANCE | | **id\_class\_instance** | | **id\_class \*** | | **id\_staff \*** | | start\_time | | end\_time | | frequency | |
| |  | | --- | | USERS | | **id\_user** | | login | | password | | profile | | |  | | --- | | PAYMENTS | | **id\_payment** | | **id\_member \*** | | date | | amount | | details | | |  | | --- | | EQUIPMENT | | **id\_equipment** | | name | | description | | id\_set | |

## Initial Entity Relationship Diagram



## Chen Diagram



# Chen Diagram V2.0

## Database Schema

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | STAFF |  | | **id\_staff** |  | | **id\_user \*** |  | | firstName | 1 | | lastName | | birthdate |  | | address\_1 | 2 | | city | | county | | postalcode | | natinsnumber | 3 | | contract\_type | | position | | qualifications | | contract\_start | | contract\_finish | | address\_2 | 4 | | emerg\_contact\_name | | emerg\_contact\_telephone | | emerg\_contact\_relation | | nationality |  | | allergies |  | | medicalNotes |  | | |  |  | | --- | --- | | MEMBERS |  | | **id\_member** |  | | **id\_user \*** |  | | firstName | 1 | | lastName | | birthDate |  | | address\_1 | 2 | | city | | county | | postalCode | | type |  | | payment\_method |  | | is\_active |  | | address\_2 | 3 | | emerg\_contact\_name | | emerg\_contact\_relation | | emerg\_contact\_phone | | emerg\_contact\_mobile | | allergies |  | | medical\_notes |  | | picture |  | | |  | | --- | | CLASSES | | **id\_class** | | name | | description | | type |  |  | | --- | | CLASS\_BOOKINGS | | **id\_class\_booking** | | **id\_class\_instance \*** | | **id\_member \*** | | booking\_date |  |  | | --- | | CLASS\_INSTANCE | | **id\_class\_instance** | | **id\_class \*** | | **id\_staff \*** | | **id\_room \*** | | date | | start\_time | | end\_time | | frequency | |
| |  | | --- | | STAFF\_EQUIPMENT\_BOOKINGS | | **id\_staff\_eq\_booking** | | **id\_staff \*** | | **id\_equipment \*** | | date\_start | | date\_due | | |  | | --- | | PAYMENTS | | **id\_payment** | | **id\_member \*** | | date | | amount | | details | | |  | | --- | | EQUIPMENT | | **id\_equipment** | | name | | description | | id\_set | |
| |  | | --- | | MEMBERS\_EQUIPMENT\_BOOKINGS | | **id\_mbr\_eq\_booking** | | **id\_member \*** | | **id\_equipment \*** | | date\_start | | date\_due | | |  | | --- | | USERS | | **id\_user** | | login | | password | | profile | | active | | |  | | --- | | ROOMS | | **id\_room** | | name | | description | | size | |

## Initial Entity Relationship Model



## Chen diagram



# Chen Diagram V3.0

## Database Schema

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | STAFF |  | | **id\_staff** |  | | **id\_user \*** |  | | firstName | 1 | | lastName | | birthdate |  | | address\_1 | 2 | | city | | county | | postalcode | | phone !!! | | mobile !!! | | natinsnumber | 3 | | contract\_type | | position | | qualifications | | contract\_start | | contract\_finish | | address\_2 | 4 | | emerg\_contact\_name | | emerg\_contact\_telephone | | emerg\_contact\_relation | | nationality |  | | medical\_allergies | 5 | | medical\_notes | | |  |  | | --- | --- | | MEMBERS |  | | **id\_member** |  | | **id\_user \*** |  | | firstName | 1 | | lastName | | birthDate |  | | address\_1 | 2 | | city | | county | | postalCode | | email | | phone !!! | | mobile !!! | | type |  | | payment\_method |  | | is\_active |  | | address\_2 | 3 | | emerg\_contact\_name | | emerg\_contact\_relation | | emerg\_contact\_phone | | emerg\_contact\_mobile | | medical\_allergies | 4 | | medical\_notes | | medical\_doc\_name | | medical\_phone | | picture |  | | member\_number |  | | |  | | --- | | CLASSES | | **id\_class** | | name | | description | | type |  |  | | --- | | CLASS\_BOOKINGS | | **id\_class\_booking** | | **id\_class\_instance \*** | | **id\_member \*** | | booking\_date |  |  | | --- | | CLASS\_INSTANCE | | **id\_class\_instance** | | **id\_class \*** | | **id\_staff \*** | | **id\_room \*** | | date | | start\_time | | end\_time | | frequency | |
| |  | | --- | | PAYMENTS | | **id\_payment** | | **id\_member \*** | | date | | amount | | details | |  | |  | | --- | | EQUIPMENT | | **id\_equipment** | | name | | description | | id\_set | |
| |  | | --- | | EQUIPMENT\_BOOKINGS | | **id\_staff\_eq\_booking** | | **id\_equipment \*** | | **id\_staff \*** | | **id\_member \*** | | **id\_class\_instance \*** | | date\_start | | date\_due | | |  | | --- | | USERS | | **id\_user** | | login | | password | | profile | | active | | |  | | --- | | ROOMS | | **id\_room** | | name | | description | | size | |

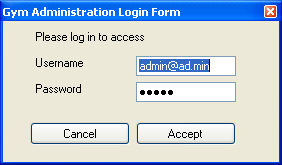
## Initial Entity Relationship Model



## Chen diagram



# Software Functions



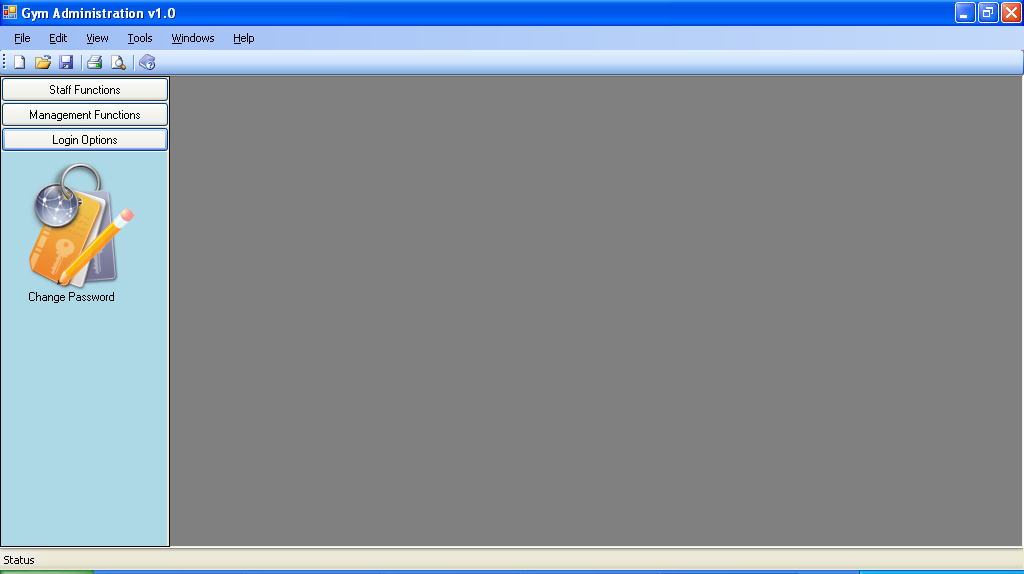
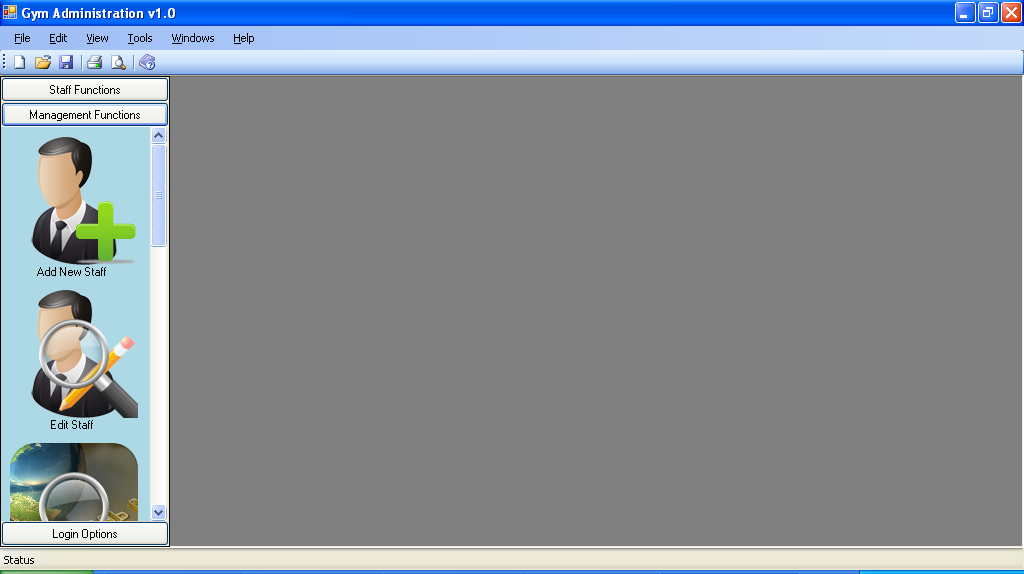
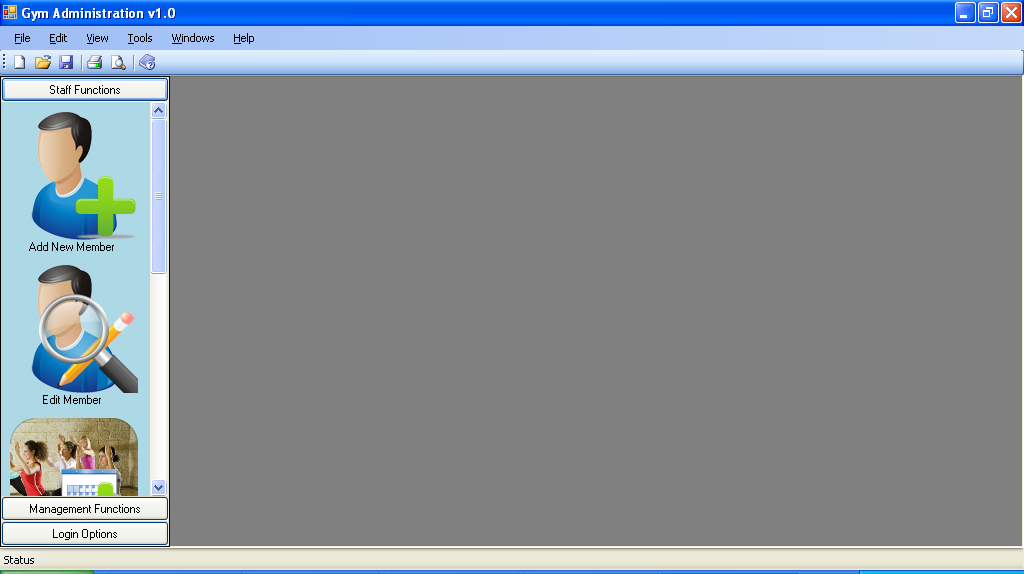
Staff Functions

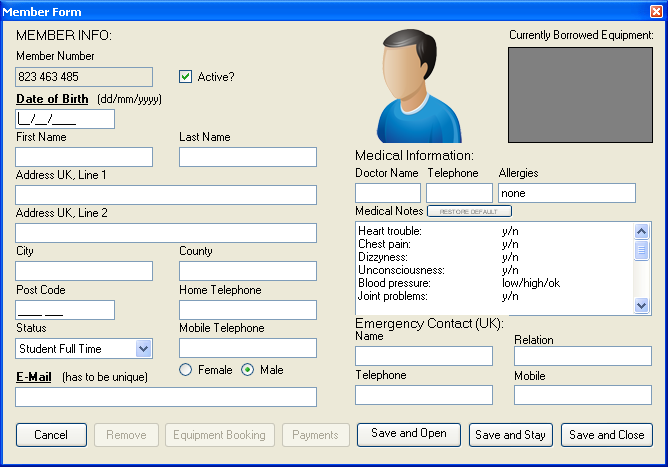
Login Options

Management Functions

Login Window

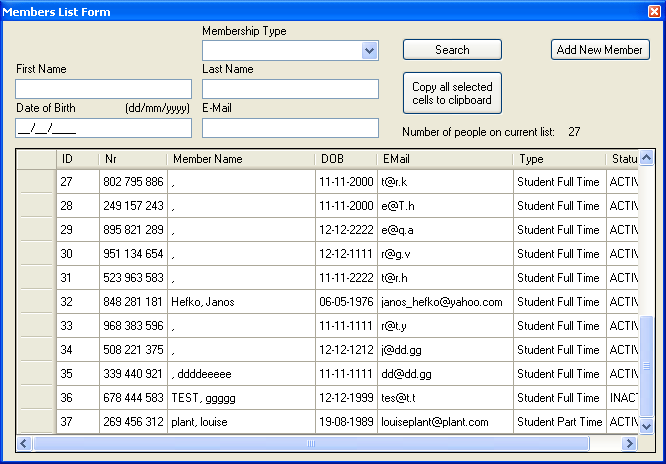
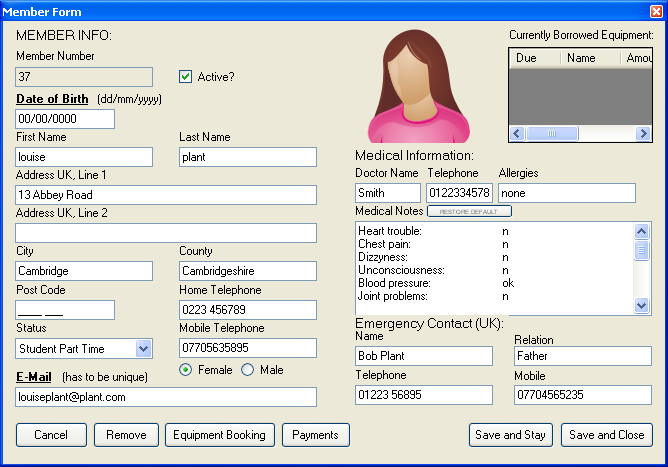
Menu





Staff Functions /Add New Member

Click Add New Member to open a blank Member Form



By filling in the fields and clicking search it makes it easier and quicker to find Members

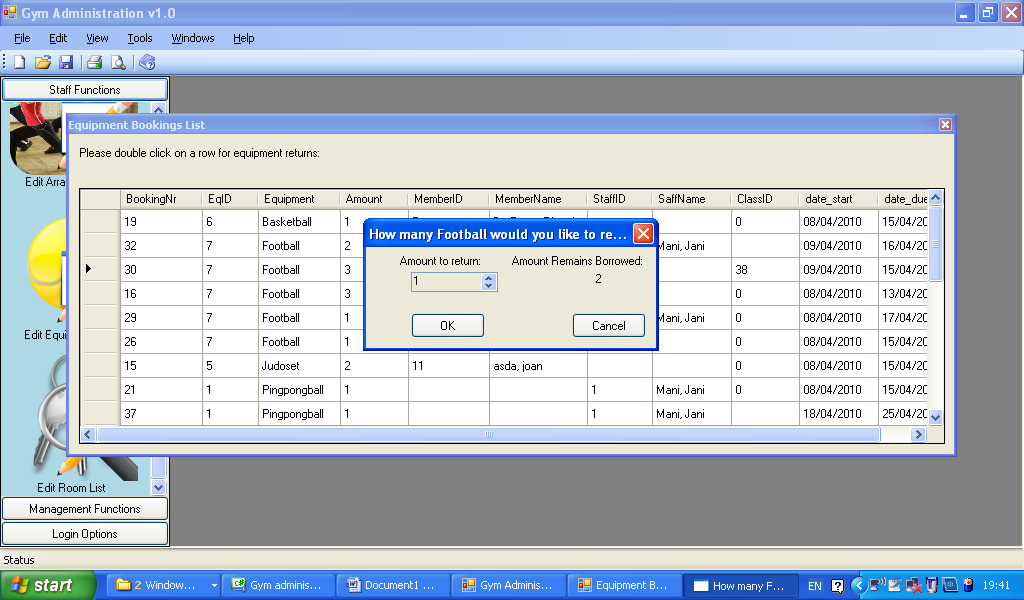
Save the information and then open the Member List

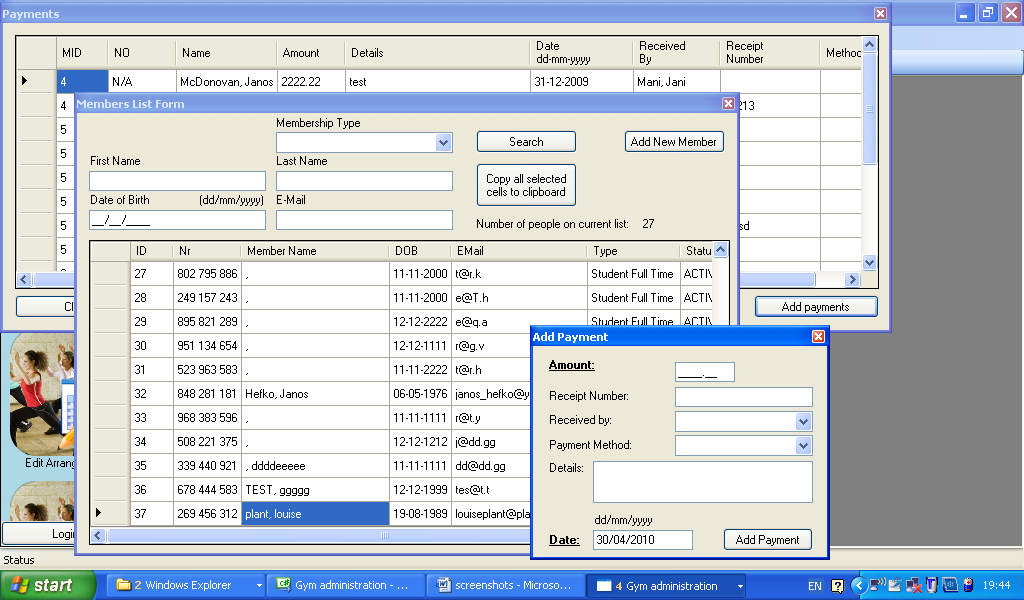
Double click a record to view the Members Information

Staff Functions /Edit Member/View Member Information/Equipment Bookings

Staff Functions /Edit Member/Member List

Equipment Bookings/Return Items





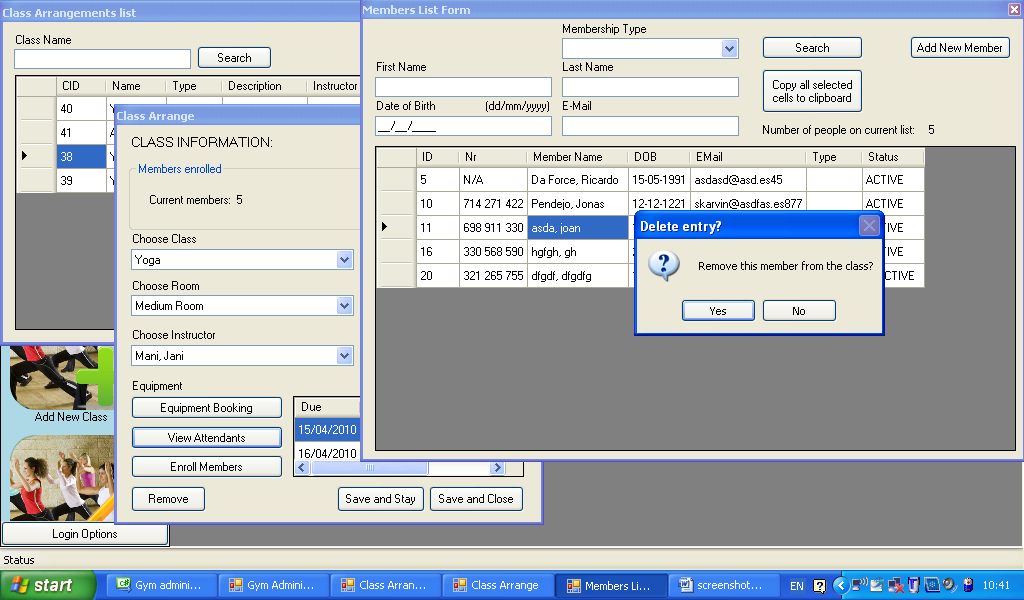
Payments List/Add Payments

Double click a booking

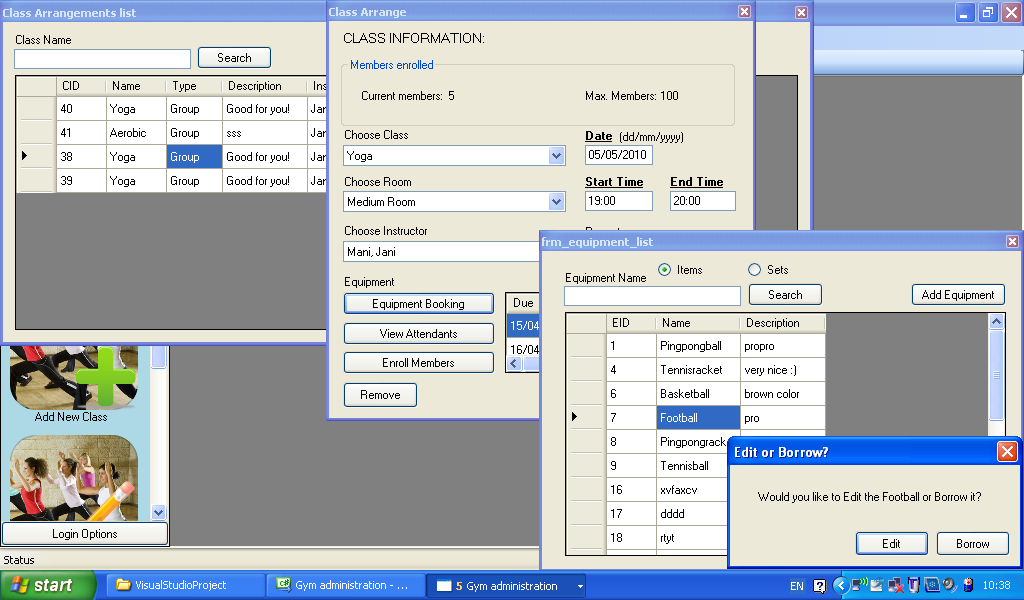
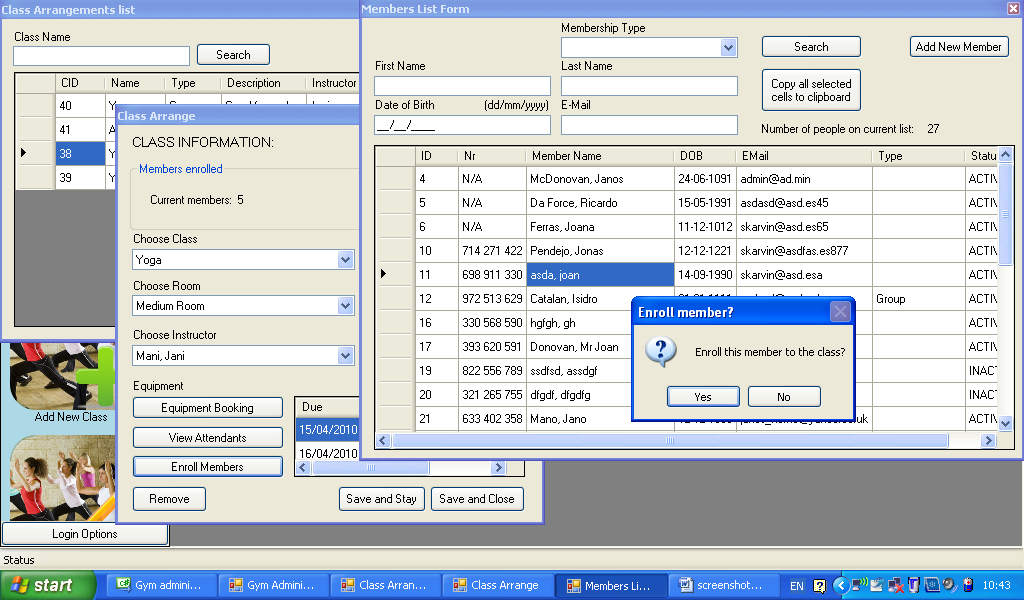
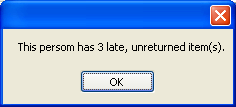
to return equipment.

Click ‘add payments’ brings up the Member List so that the user can search for a Member.

Then double click the user that has made the payment to record the information.



Click to bring up the list of members attending that class.



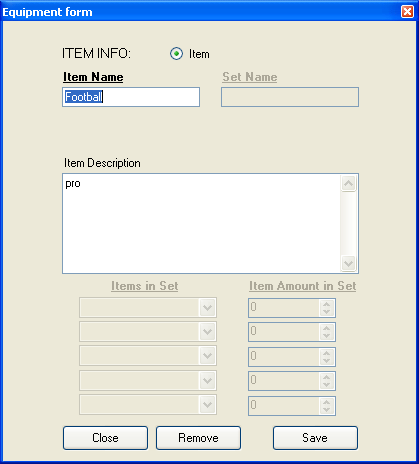
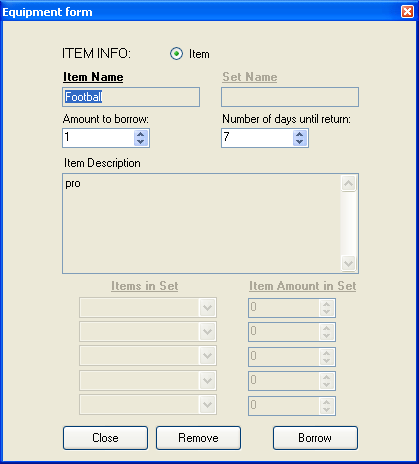
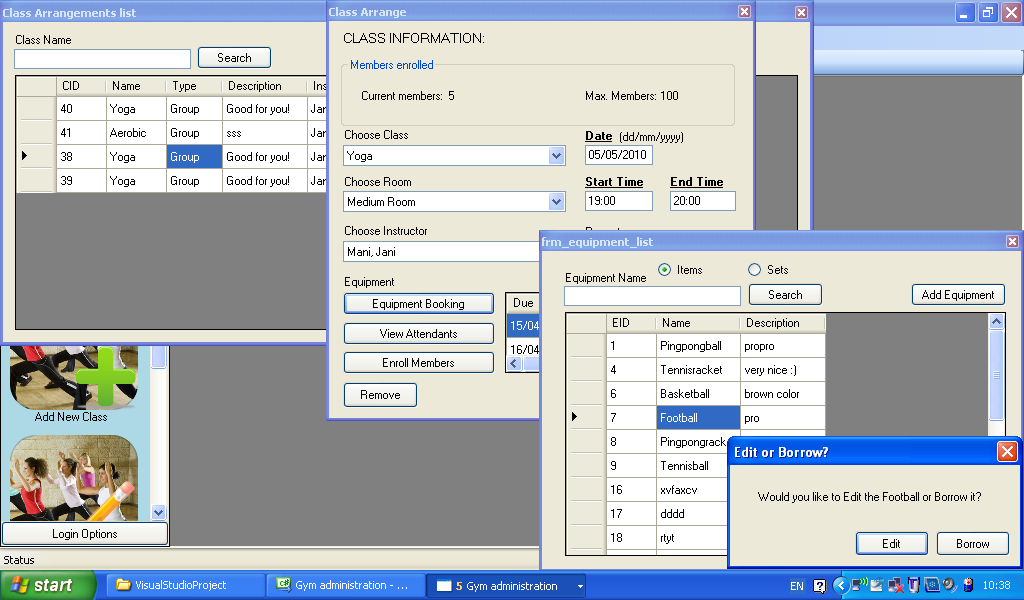
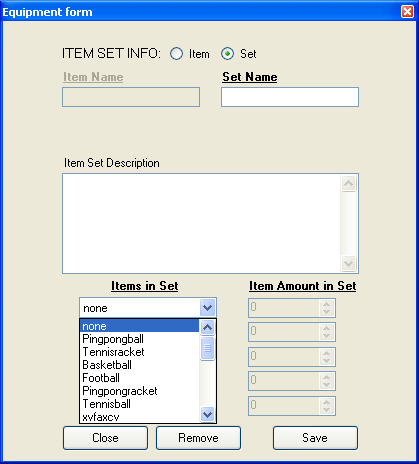
Once the user confirms their choice then a notice is displayed if they have equipment that needs returning.

Double click a Member you wish to enroll on the class.

Double click item or set to display options.

Add a Equipment Booking to this class.

Double click a member to remove them from the class.

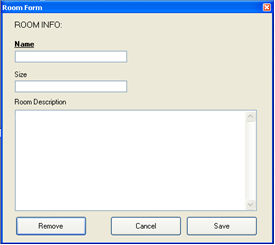


Staff Functions/Equipment List/Edit Equipment/Borrow Equipment/Add Equipment

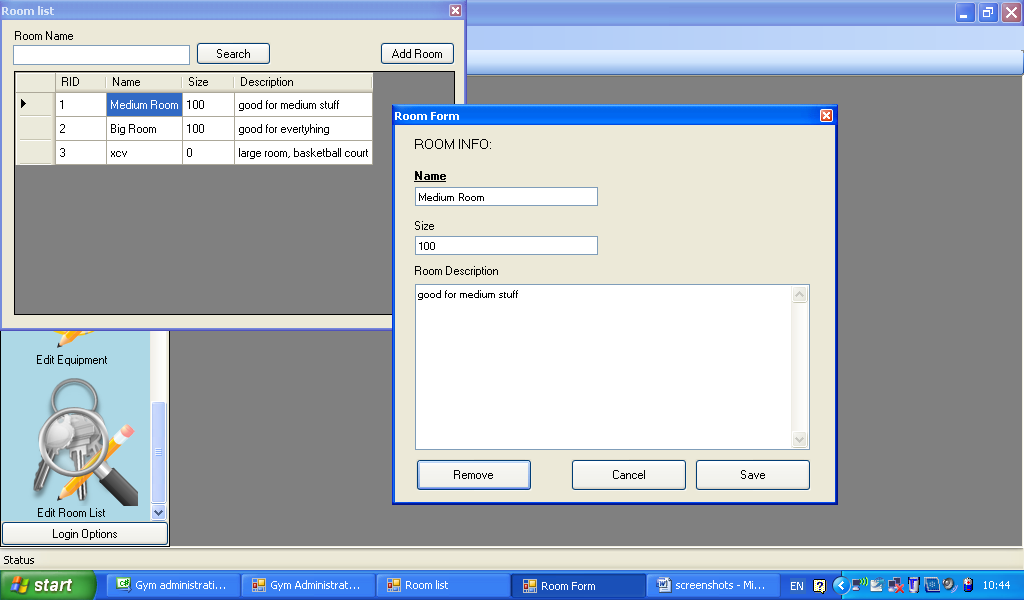
Adding equipment brings up a plant Equipment Form that the user can fill out and then save, the equipment will then appear on the Equipment List.

Borrowing equipment freezes the text fields and brings up the option for users to choose how many pieces of equipment they need.

Editing the equipment lets the user change the information stored in the text fields and then save those changes.

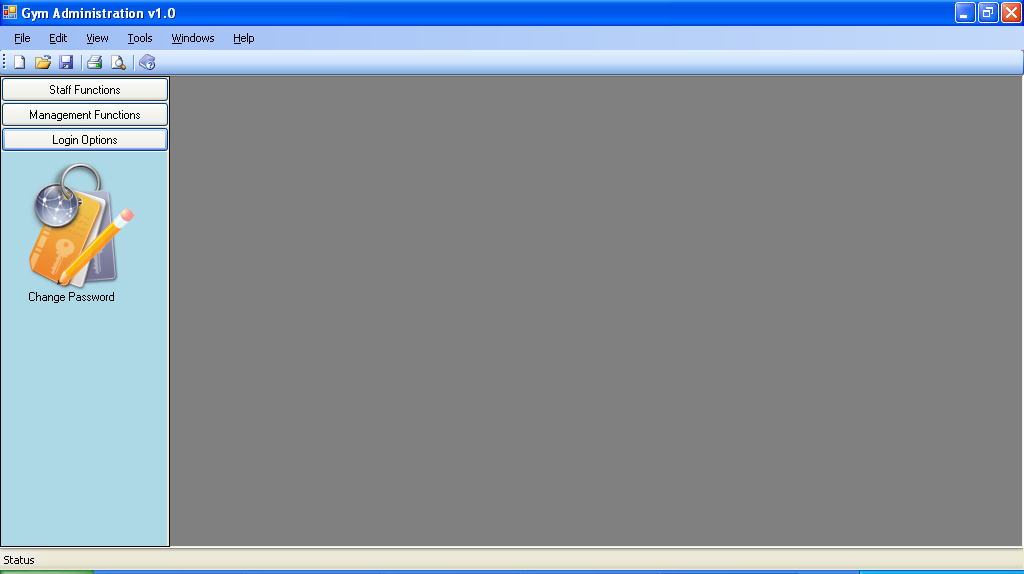
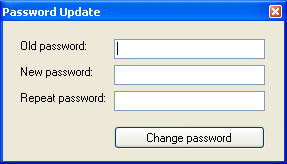


Clicking Add Room will open a empty Room Form that can then be filled in and saved. The room will then appear on the Room List.



Management Functions/Room List/Edit Room/Add Room

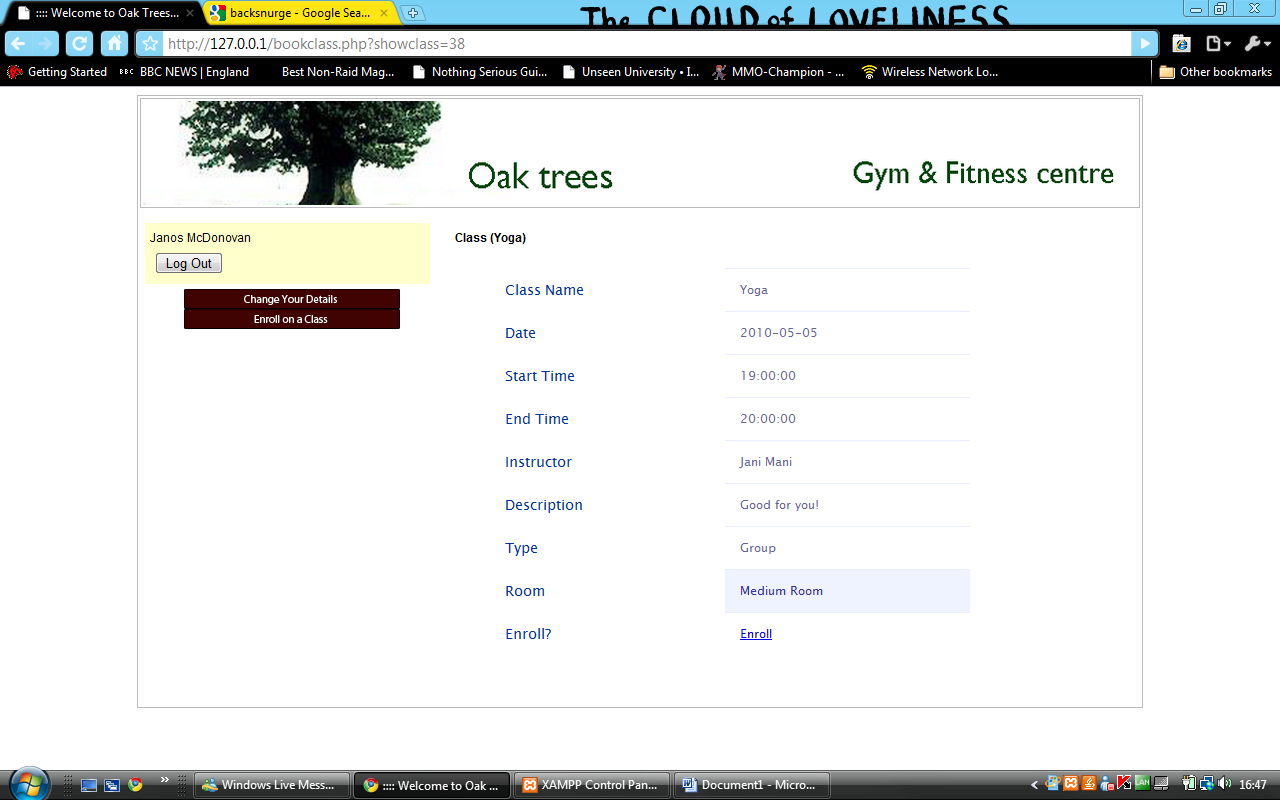
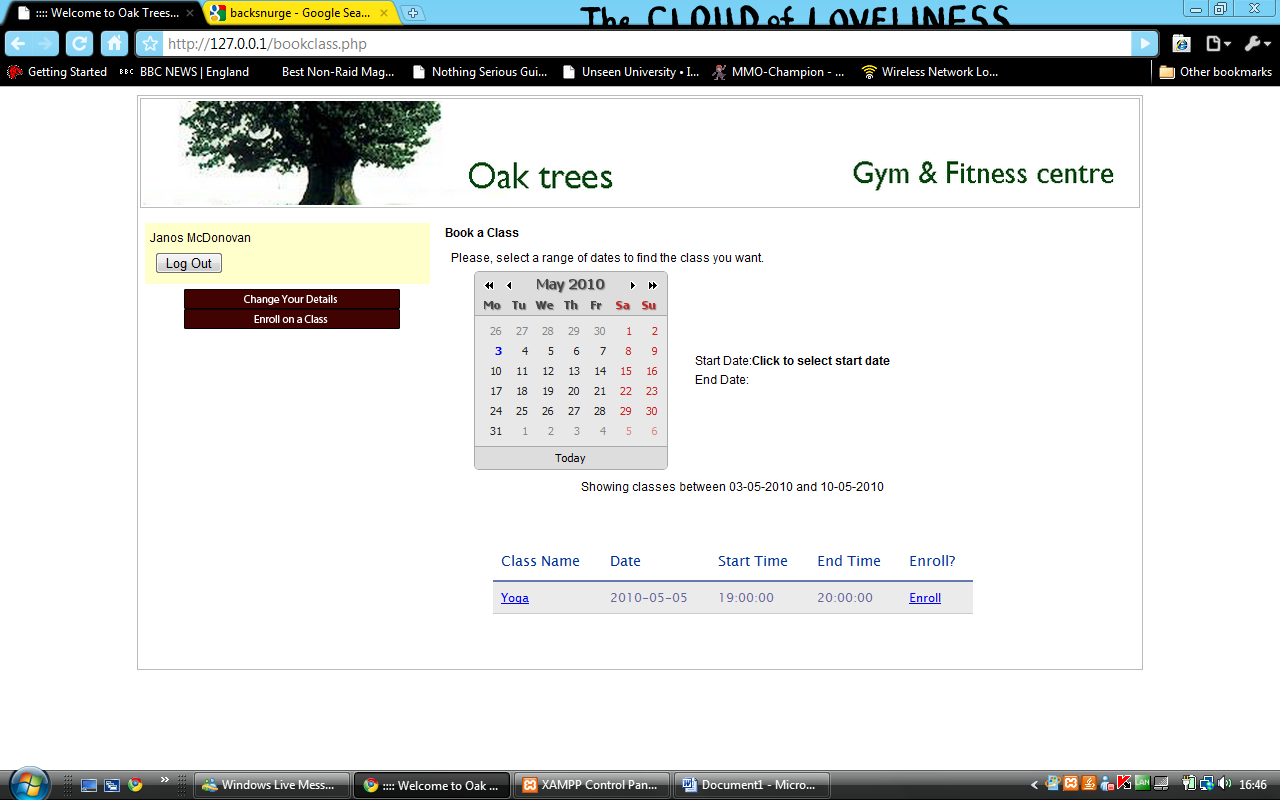
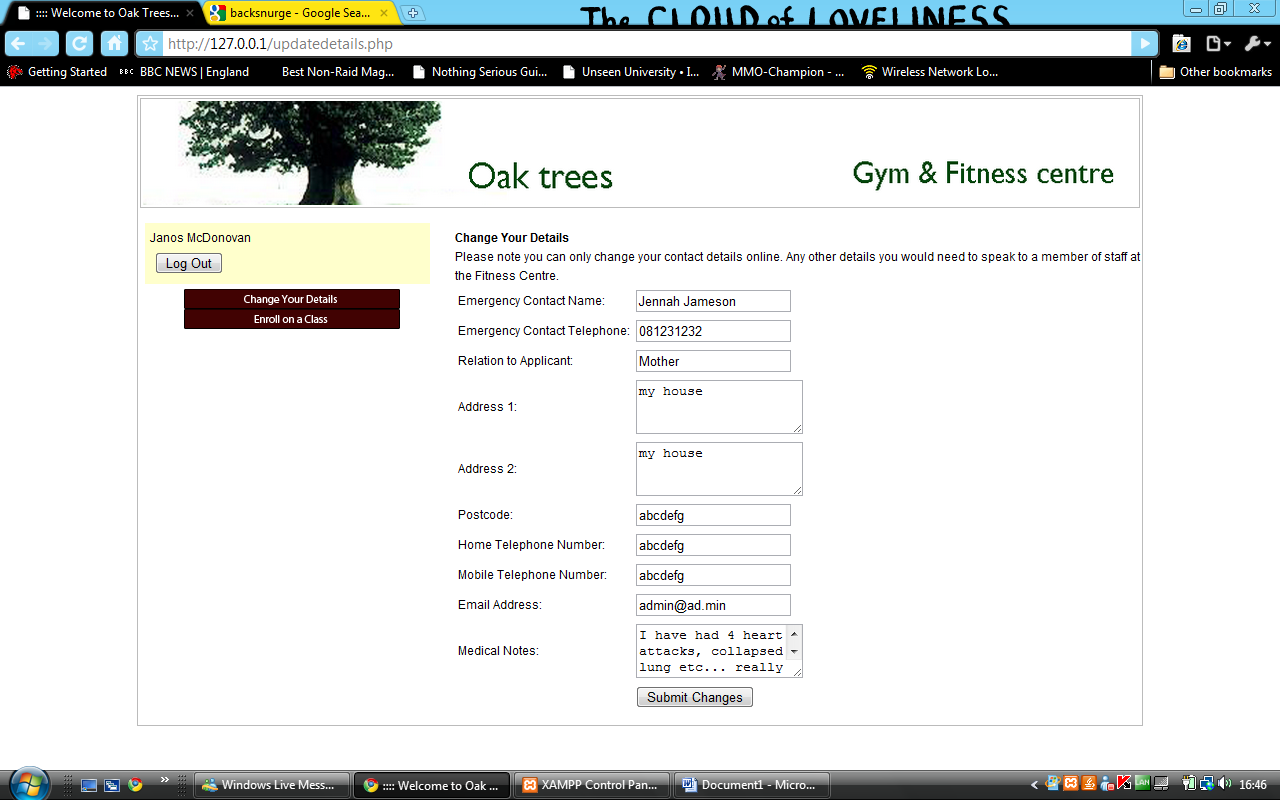
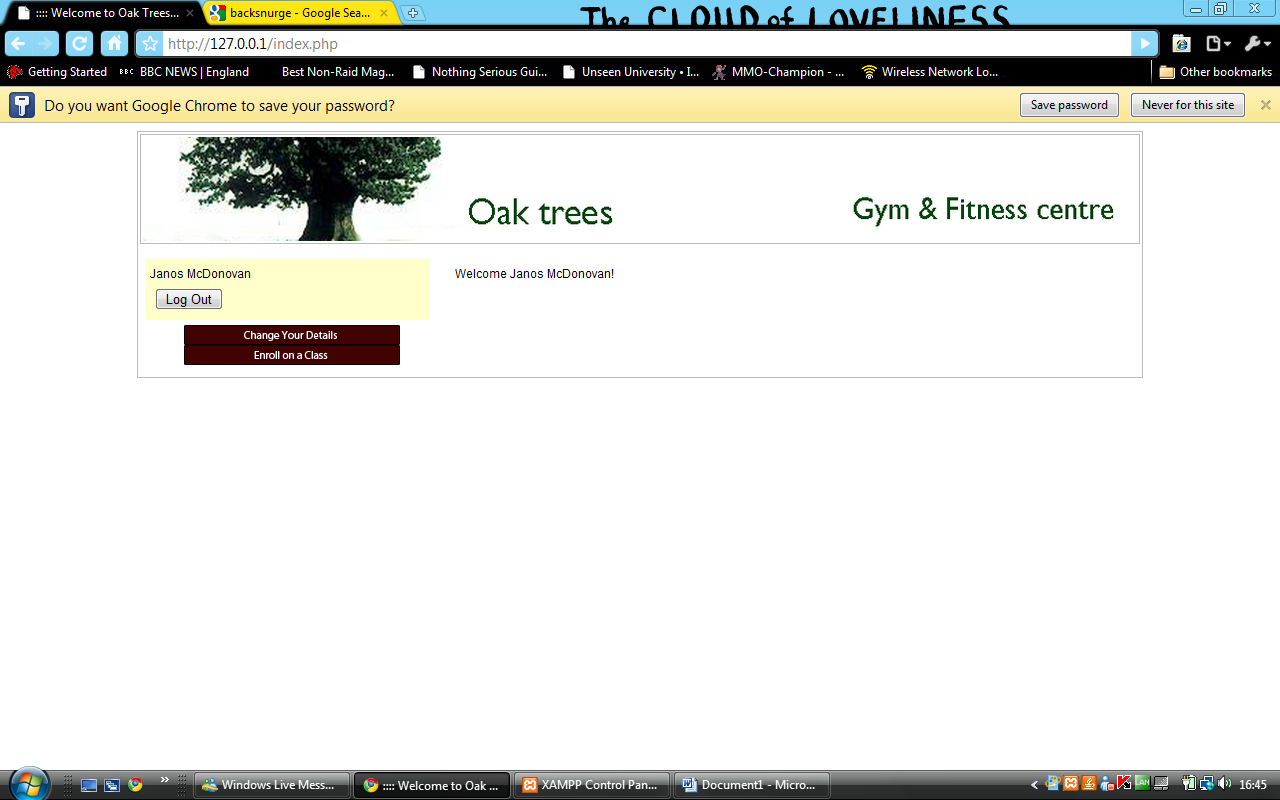
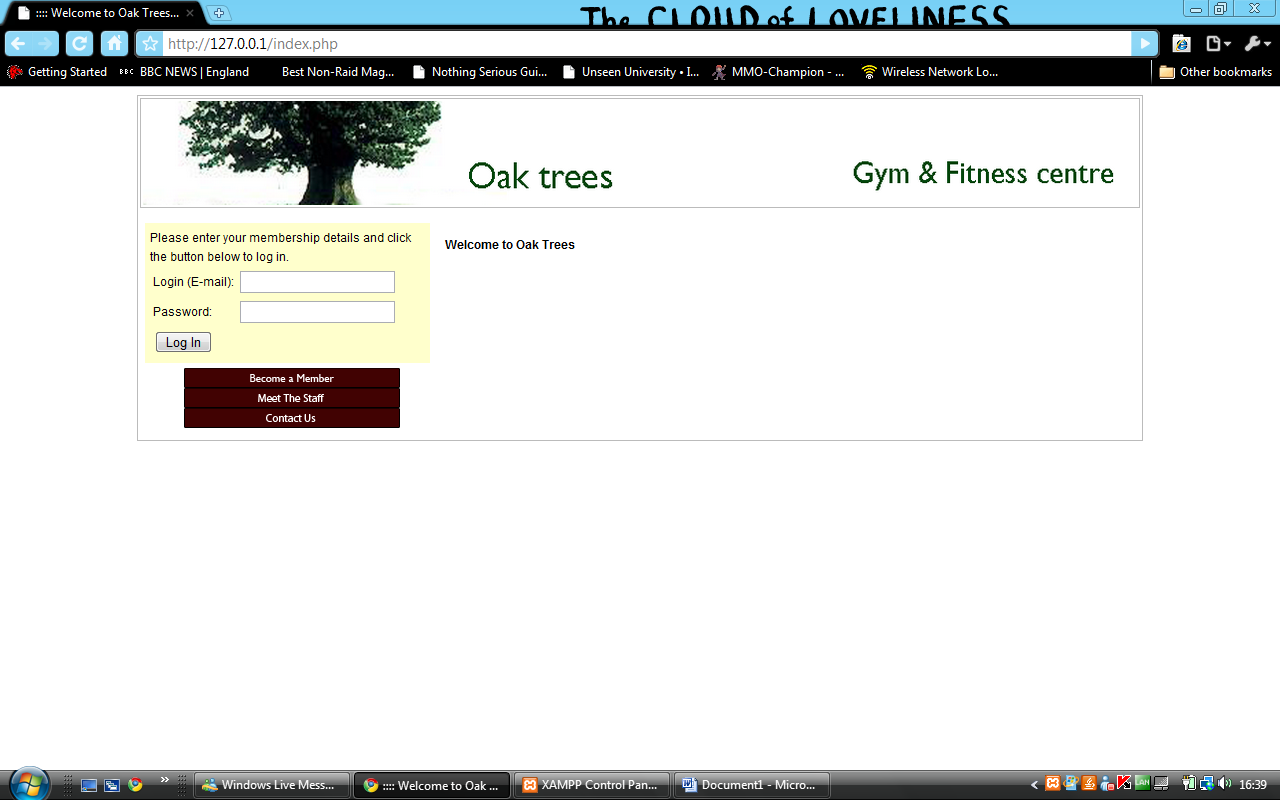
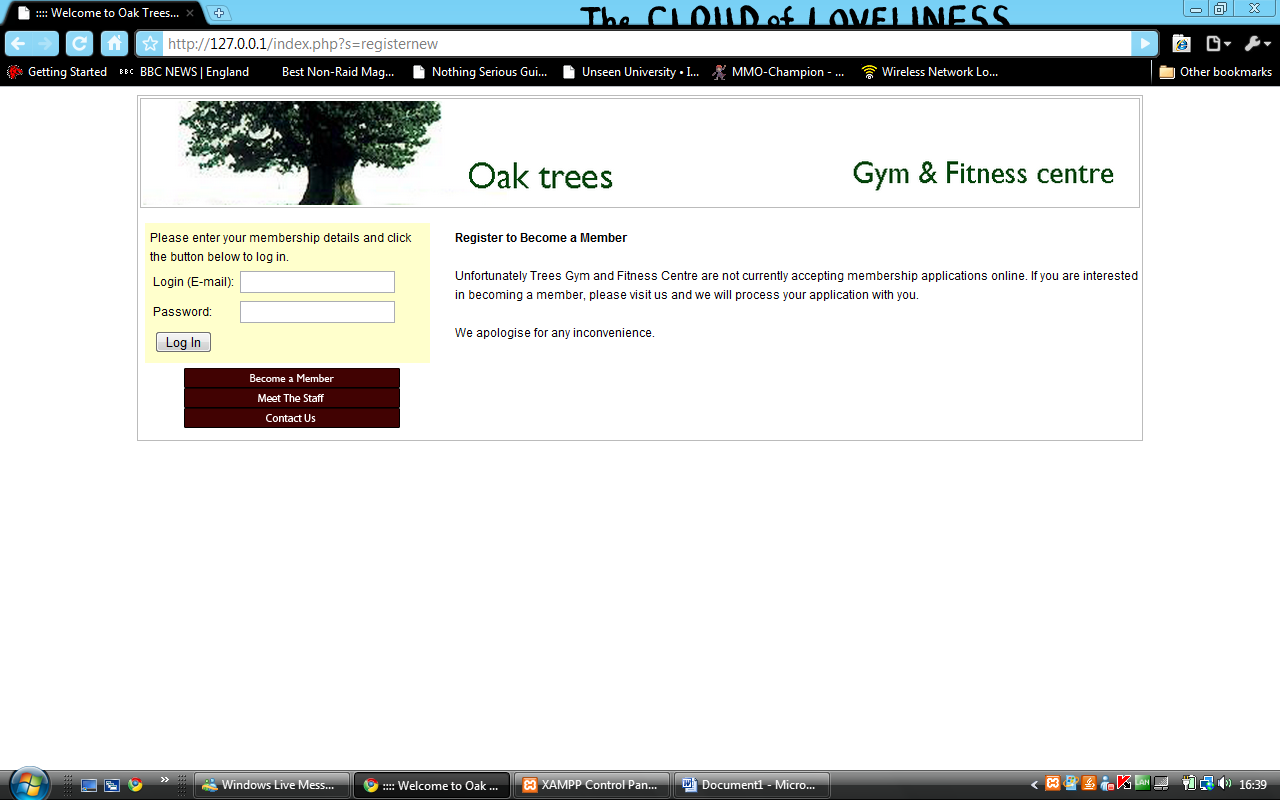
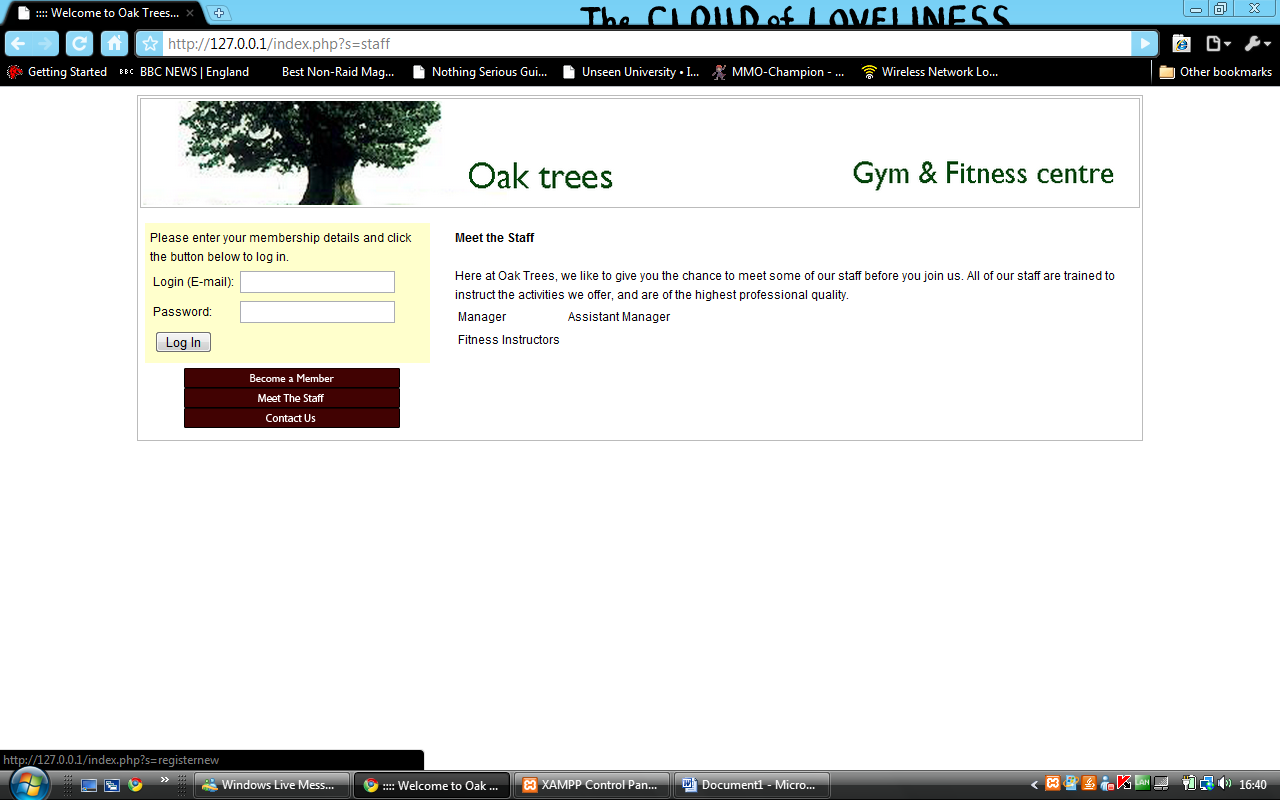
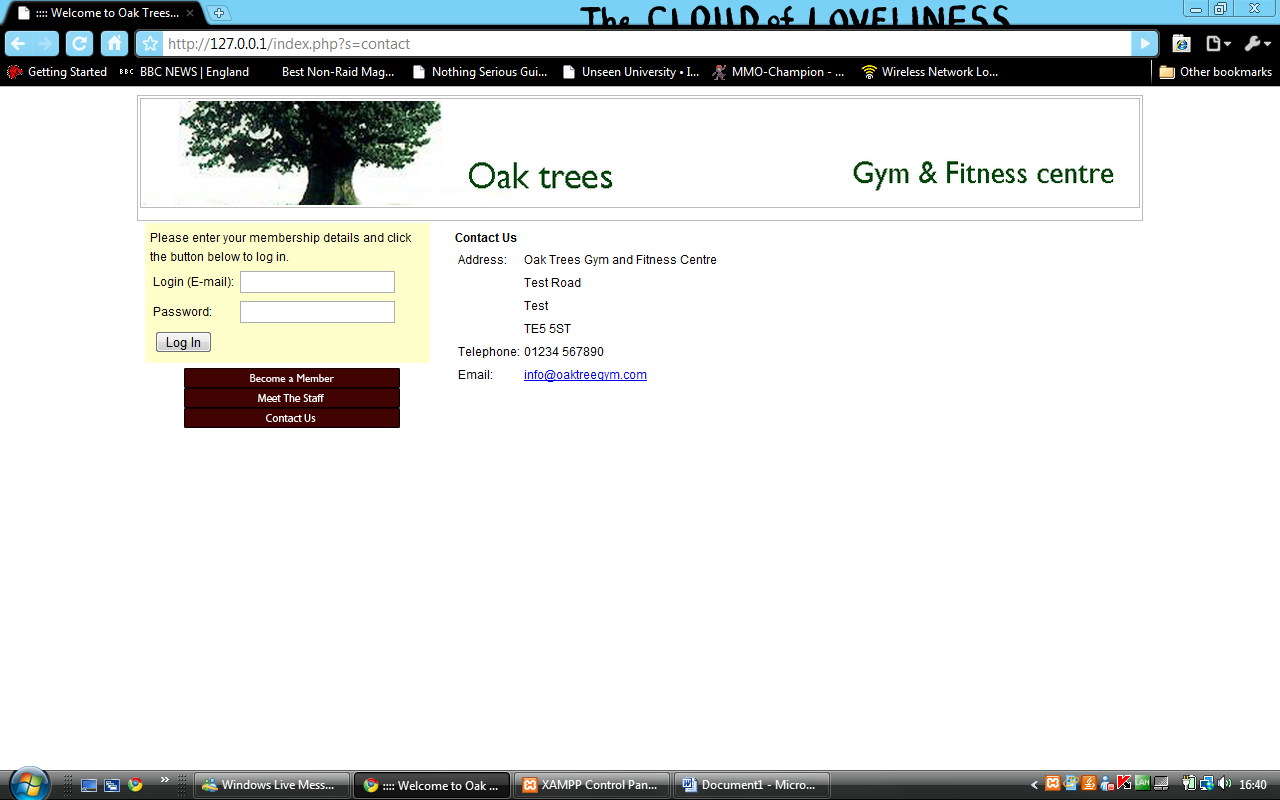
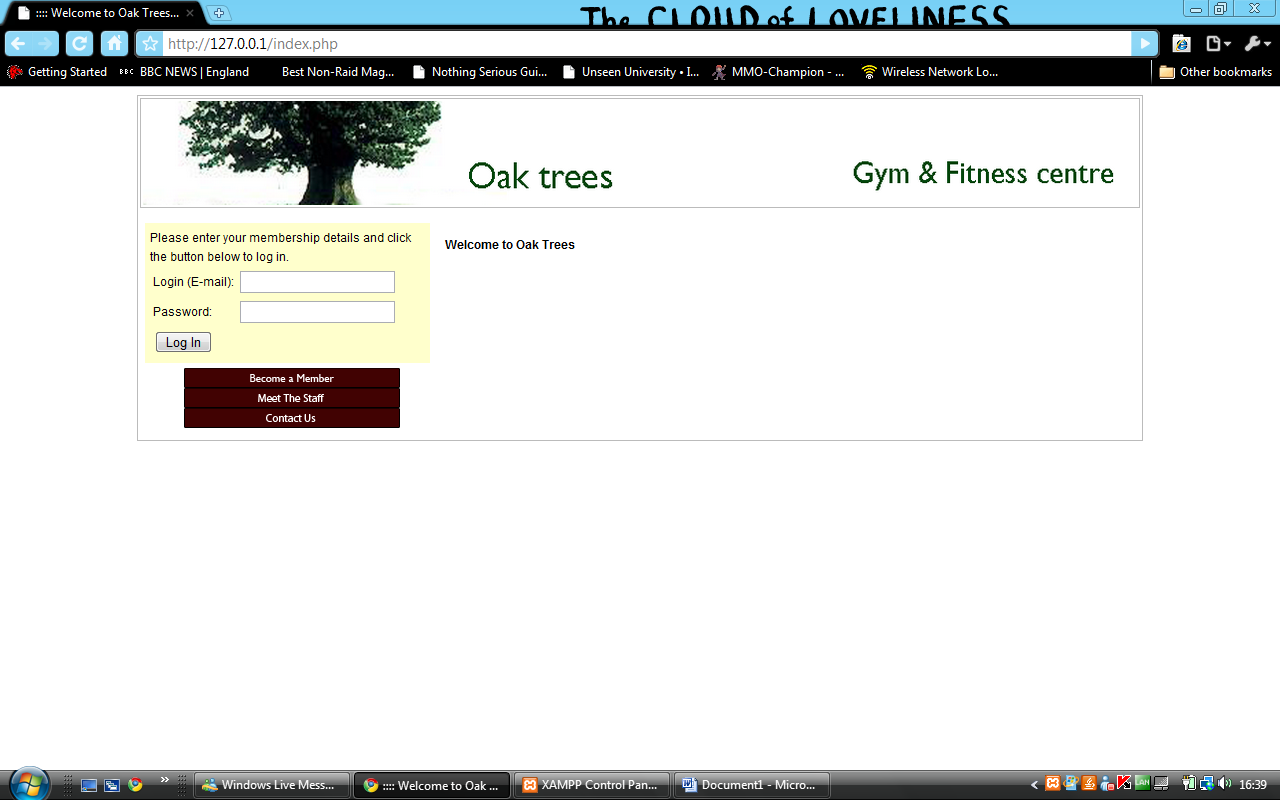
Double Click a room form the Room List and the information can then edited and saved. The room can also be removed from the list by clicking the button on the bottom left hand side.



Once logged in the user can choose this option to change their password.

Login Options/Change Password

# Website Functions

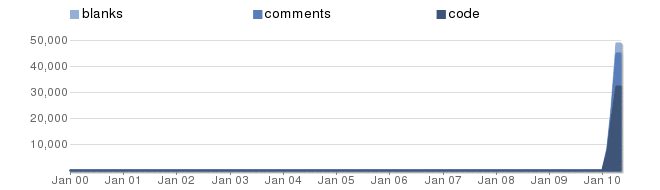


# Coding

Ohloh analyzes the project source code and determines the language of each line of code, excluding comments and blanks.

|  |  |  |
| --- | --- | --- |
| C# | 57% | anguages |
| PHP | 18% |
| XML | 12% |
| OTHER | 13% |

**Lines of Code**



**Lines of Code By Language**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Language** | **Code Lines** | **Comment Lines** | **Comment Ratio** | **Blank Lines** | **Total Lines** |
|  | [C#](https://www.ohloh.net/languages/17) | 18,458 | 7,448 | 28.8% | 1,871 | 27,777 |
|  | [PHP](https://www.ohloh.net/languages/2) | 5,889 | 3,192 | 35.2% | 1,325 | 10,406 |
|  | [XML](https://www.ohloh.net/languages/3) | 4,009 | 1,934 | 32.5% | 440 | 6,383 |
|  | [CSS](https://www.ohloh.net/languages/4) | 1,431 | 19 | 1.3% | 209 | 1,659 |
|  | [HTML](https://www.ohloh.net/languages/1) | 1,262 | 1 | 0.1% | 57 | 1,320 |
|  | [JavaScript](https://www.ohloh.net/languages/6) | 823 | 29 | 3.4% | 169 | 1,021 |
|  | [SQL](https://www.ohloh.net/languages/30) | 433 | 17 | 3.8% | 9 | 459 |

ClassInstance.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections;

using System.Windows.Forms;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the CLASS\_INSTANCE table.

\* Which is about a single class instance (certain class at a certain time).

\* Most closely associated form is frm\_class\_instance\_arrange,

\* when frm\_class\_instance\_arrange is called from frm\_class\_instance\_list.

\* Used also in frm\_member\_list, as frm\_member\_list is used to book attendants for a class instance.

\* Most closely associated table is CLASS\_INSTANCE.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

public class ClassInstance

{

// id\_class\_instance field from CLASS\_INSTANCE table stored here

private int id\_class\_instance;

public int Id\_class\_instance

{

get { return id\_class\_instance; }

set { id\_class\_instance = value; }

}

// a 'Gym class' (Class.cs) object is stored here

private Class clClass;

internal Class ClClass

{

get { return clClass; }

set { clClass = value; }

}

// id\_staff field from CLASS\_INSTANCE table stored here

private int id\_staff;

public int Id\_staff

{

get { return id\_staff; }

set { id\_staff = value; }

}

// date field from CLASS\_INSTANCE table stored here

private string dateStart;

public string DateStart

{

get { return dateStart; }

set { dateStart = value; }

}

// start\_time field from CLASS\_INSTANCE table stored here

private string startTime;

public string StartTime

{

get { return startTime; }

set { startTime = value; }

}

// end\_time field from CLASS\_INSTANCE table stored here

private string endTime;

public string EndTime

{

get { return endTime; }

set { endTime = value; }

}

// frequency field from CLASS\_INSTANCE table stored here

private string frequency;

public string Frequency

{

get { return frequency; }

set { frequency = value; }

}

// a Room (Room.cs) object is stored here

private Room clRoom;

internal Room ClRoom

{

get { return clRoom; }

set { clRoom = value; }

}

// A list of member objects participating in the class instance stored here

private List<Member> lclAttendants = new List<Member>();

internal List<Member> LclAttendants

{

get { return lclAttendants; }

set { lclAttendants = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_class to -1 so the fact of this is a new class instance can be referenced.

\*

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public ClassInstance()

{

this.id\_class\_instance = -1;

}

/\*\*

\* @desc Constructor.

\* Loads in various info from tables CLASSES, CLASS\_INSTANCE and STAFF for this class instance.

\* Loads in all atendants for this class instance.

\* @params [int] id\_class\_instance identifies the class instance uniquely.

\* @return [none] No directly returned data.

\*/

public ClassInstance(int id\_class\_instance)

{

// Create mysql connection.

mySqlConn conn = new mySqlConn();

conn.connect();

// Launch the query to return all all fields from a single class instance row of the CLASS\_INSTANCE table.

List<Hashtable> lhResultSet = conn.lhSqlQuery("SELECT ci.id\_class\_instance, c.name, c.type, c.description, s.firstName, s.id\_staff, DATE\_FORMAT(ci.date, '%d/%m/%Y') date, ci.start\_time, ci.end\_time, ci.id\_room, c.id\_class, ci.frequency FROM classes c, class\_instance ci, staff s WHERE ci.id\_class = c.id\_class AND ci.id\_staff = s.id\_staff AND ci.id\_class\_instance = '" + id\_class\_instance + "'");

// Check if we found the row

if ((int)lhResultSet.Count > 0)

{

// Fill in all class instance fields with table data

this.Id\_class\_instance = int.Parse(lhResultSet[0]["id\_class\_instance"].ToString());

this.Id\_staff = int.Parse(lhResultSet[0]["id\_staff"].ToString());

this.ClRoom = new Room(int.Parse(lhResultSet[0]["id\_room"].ToString()));

this.ClClass = new Class(int.Parse(lhResultSet[0]["id\_class"].ToString()));

this.DateStart = lhResultSet[0]["date"].ToString();

this.EndTime = lhResultSet[0]["end\_time"].ToString();

this.StartTime = lhResultSet[0]["start\_time"].ToString();

this.Frequency = lhResultSet[0]["frequency"].ToString();

// Create a list for storing member objects

// Load in all records for the same class instance from CLASS\_BOOKINGS table (each contains a different member ID)

List<Hashtable> lhResultSetBookings = conn.lhSqlQuery("SELECT \* FROM `gym`.`class\_bookings` WHERE id\_class\_instance = '" + id\_class\_instance + "'");

// If there is any class booking (any member enrolled) exist with the class instance id

if ((int)lhResultSetBookings.Count > 0)

{

// Create a list of attending members

foreach (Hashtable hClassBooking in lhResultSetBookings)

{

// Retrieve the member number from the current class booking

int id\_member = int.Parse(hClassBooking["id\_member"].ToString());

// Create a corresponding member object with all the particular member info loaded into it

Member clMember = new Member(id\_member);

// If a member with this id\_member actually exist, then add the member object to the list

if(clMember.Id\_member != -1)

this.lclAttendants.Add(clMember);

}

}

}

}

/\*\*

\* @desc Method for checking if there is any overlap of class instances in the same room or same instructor at conflicting times

\* @params [string] sDate has the date of the class instance

\* @params [string] id\_room is the room where the class instance takes place

\* @params [string] id\_staff is the instructor on this occasion

\* @params [string] sStartTime is the start time

\* @params [string] sEndTime is the end time

\* @return [bool] Returns true if there is an overlap and false if everything is green ligth

\*/

public bool bCheckOverlap(string date, string id\_room, string id\_staff, string dtartTime, string endTime)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create the overlap check query

string query = "SELECT \* FROM gym.class\_instance WHERE date = '" + date + "' AND (id\_room = '" + id\_room + "' OR id\_staff = '" + id\_staff + "') AND (" +

"(start\_time BETWEEN '" + startTime + "' AND '" + endTime + "') OR " +

"(end\_time BETWEEN '" + startTime + "' AND '" + endTime + "') OR " +

"(start\_time < '" + startTime + "' AND end\_time > '" + endTime + "') OR " +

"(start\_time > '" + startTime + "' AND end\_time < '" + endTime + "'))" + ((this.Id\_class\_instance != -1)?" AND id\_class\_instance != '"+this.Id\_class\_instance+"'":"");

// Launch the overlap check query and load the result into a hashtable

List<Hashtable> lhResultSet = conn.lhSqlQuery(query);

// If there is any result then there is an overlap

if ((int)lhResultSet.Count >= 1)

return true;

// Otherwise ther is no overlap

return false;

}

/\*\*

\* @desc This method will save or update a class instance in the CLASS\_INSTANCE table

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem saving/updating the class

\*/

public bool SaveClassInstance()

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// SAVING THE CLASS INSTANCE INTO CLASS\_INSTANCE

// Check whether there is a new id\_class\_instance assigned to this class instance,

// if not then this a new class to save

if (this.Id\_class\_instance == -1)

{

// Create the save query

string saveClInstanceQuery = "insert into `gym`.`class\_instance` (`id\_class\_instance`, `id\_class`, `id\_staff`, `date`, `start\_time`, `end\_time`, `frequency`, `id\_room`) values " +

"(NULL, '" + this.ClClass.Id\_class + "', '" + this.Id\_staff + "', '" + Utils.sGetMysqlDate(this.DateStart) + "', '" + this.StartTime + "', '" + this.EndTime + "', '" + this.Frequency + "', '" + this.ClRoom.Id\_room + "');";

// Launch save query

int id\_class\_instance = conn.InsertToDB(saveClInstanceQuery);

// Check saving result

if (id\_class\_instance != -1)

{

this.Id\_class\_instance = id\_class\_instance;

MessageBox.Show("The class has been saved!");

return true;

}

else

{

MessageBox.Show("There has been an error creating the class instance! Contact with your administrator.");

}

}

// If an id\_class\_instance already exists for this class instance, then this is an existing class instance to update

else

{

string updateClInstanceQuery = "UPDATE class\_instance SET id\_staff= '" + this.Id\_staff + "', date = '" + Utils.sGetMysqlDate(this.DateStart) + "', start\_time = '" + this.StartTime + "', end\_time = '" + this.EndTime + "', frequency = '" + this.Frequency + "', id\_room = '" + this.ClRoom.Id\_room + "' " +

"WHERE id\_class\_instance = '" + this.Id\_class\_instance + "'";

// Launch update query

int result = conn.DeleteOrUpdate(updateClInstanceQuery);

// Check update result

if (result > 0)

{

MessageBox.Show("The class booking data has been updated succesfully!");

}

else

{

MessageBox.Show("There was a problem updating the class booking information, please check your data!");

return false;

}

}

// SAVING THE ATTENDANTS INTO CLASS\_BOOKINGS

// If there are any attendants save them into the CLASS\_BOOKINGS table

if (this.lclAttendants.Count > 0)

{

StringBuilder sbQueryClassBooking = new StringBuilder();

// Create the first half of the insert query containing so far only the fields

sbQueryClassBooking.Append("insert into `gym`.`class\_bookings` (`id\_class\_booking`, `id\_member`, `id\_class\_instance`, `booking\_date`) values ");

int i = 0;

// Create a string array storing each second half of the queries, that is the values of each booking

string[] aQueryClassBookingValues = new string[this.lclAttendants.Count];

// Copy all values of each booking into each string in the array

foreach (Member clMember in this.lclAttendants)

{

aQueryClassBookingValues[i] = "(NULL, '" + clMember.Id\_member + "', '" + this.Id\_class\_instance + "', NOW())";

i++;

}

// It is possible to add more than one class booking into the CLASS\_BOOKINGS table in the same query at the same time

// so append each set of booking values one after the other at the end of the query, separated by comma

sbQueryClassBooking.Append(string.Join(", ", aQueryClassBookingValues));

sbQueryClassBooking.Append(" ON DUPLICATE KEY UPDATE booking\_date = booking\_date");

int lastMemberId = conn.InsertToDB(sbQueryClassBooking.ToString());

if (lastMemberId != -1)

{

MessageBox.Show("The attendant has been enrolled!");

}

}

return true;

}

/\*\*

\* @desc Removes the class instance from the CLASS\_INSTANCE table.

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem deleting the class instance.

\*/

public bool RemoveClassInstance()

{

if (this.Id\_class\_instance != 0)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create the delete query

string query = "DELETE FROM class\_instance WHERE id\_class\_instance = '" + this.Id\_class\_instance + "'";

// Launch delete query

int result = conn.DeleteOrUpdate(query);

// Check deletion result

if (result > 0)

{

MessageBox.Show("The class data has been deleted succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem deleting the class!");

return false;

}

}

MessageBox.Show("There is no class id!");

return false;

}

}

}

Class.cs

/ Microsoft's Capitalization Conventions from .NET Framework Developer's Guide

// http://msdn.microsoft.com/en-us/library/ms229043%28v=VS.90%29.aspx

// We tried to make the variable naming conventions to be very similar in this project:

//

// Class name:

// ClassName

// (stored in ClassName.cs)

//

// Windows Form Class name:

// frm\_windows\_form

// (stored in frm\_windows\_form.cs)

//

// Locally declared forms:

// frmWindowsForm

//

// Local primitive types:

// variableName

//

// Get/Set for primitive types:

// VariableName

//

// Arrays, Lists (type of objects are marked if locally declared class as cl (class) or h (hashtable))

// aArray, lList, lclClassList, lhHashtableList etc

//

// Get set for Arrays, Lists

// AArray, Llist, LclClassList etc

//

// Class type variables and locally declared classes:

// clClassName

//

// Get/Set for class type variables :

// ClClassName

//

// Exceptions:

// All database entity identifiers regardless of primitive type:

//

// Private/Local:

// id\_entity\_identifier

//

// Public/Global

// Id\_entity\_identifier

//

// Methos: bool Save()

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections;

using System.Windows.Forms;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the CLASSES table.

\* This is about "Gym Class" that is a general gym "class activity",

\* (NOT a certain class at a certain time!)

\* Most closely associated form is frm\_class.

\* Most closely associated table is CLASSES.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class Class

{

// id\_class field from CLASSES table stored here

private int id\_class;

public int Id\_class

{

get { return id\_class; }

set { id\_class = value; }

}

// name field from CLASSES table stored here

private string name;

public string Name

{

get { return name; }

set { name = value; }

}

// description field from CLASSES table stored here

private string description;

public string Description

{

get { return description; }

set { description = value; }

}

// type field from CLASSES table stored here

private string type;

public string Type

{

get { return type; }

set { type = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_class to -1 so the fact of this is a new class can be referenced.

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public Class()

{

this.id\_class = -1;

}

/\*\*

\* @desc Constructor

\* Loads in all fields from a single "Gym Class" row of the CLASSES table.

\* @params [int] id\_class identifies the class uniquely.

\* @return [none] No directly returned data.

\*/

public Class(int id\_class)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Launch the query to return all fields from a single "Gym Class" row of the CLASSES table

List<Hashtable> lhResultSet = conn.lhSqlQuery("Select \* from classes WHERE id\_class = '" + id\_class + "'");

// Check if we found the row

if ((int)lhResultSet.Count > 0)

{

// Fill in all class fields with table data

this.Id\_class = int.Parse(lhResultSet[0]["id\_class"].ToString());

this.Type = lhResultSet[0]["type"].ToString();

this.Description = lhResultSet[0]["description"].ToString();

this.Name = lhResultSet[0]["name"].ToString();

}

}

/\*\*

\* @desc Removes the class from the CLASSES table.

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem deleting the class.

\*/

public bool RemoveClass()

{

// Check if there is a class already loaded in

if (this.Id\_class != -1)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create the delete query

string query = "DELETE FROM classes WHERE id\_class = '" + this.Id\_class + "'";

// Launch delete query

int result = conn.DeleteOrUpdate(query);

// Check deletion result

if (result > 0)

{

MessageBox.Show("The class data has been deleted succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem deleting the class!");

return false;

}

}

return false;

}

/\*\*

\* @desc This method will save or update a class in the CLASS table

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem saving/updating the class

\*/

public bool SaveClass()

{

string query;

// Checking user input

if (this.Name == "")

{

MessageBox.Show("Please Insert a name.");

}

else

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Check whether there is a new id\_class assigned to this class,

// if not then this a new class to save

if (this.Id\_class == -1)

{

// Create the save query

query = "insert into `gym`.`classes` (`id\_class`, `name`, `type`, `description`) values " +

"(NULL, '" + this.Name + "', '" + this.Type + "', '" + this.Description + "')";

// Launch save query

int id\_class = conn.InsertToDB(query);

// Check saving result

if (id\_class != -1)

{

this.Id\_class = id\_class;

MessageBox.Show("The new class has been added to the databse succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem adding the new class, please check your data!");

return false;

}

}

// If an id\_class already exists for this class instance, then this is an existing class to update

else

{

// Create update query

query = "UPDATE classes SET name = '" + this.Name + "', description = '" + this.Description + "' " +

" WHERE id\_class = '" + this.Id\_class + "'";

// Launch update query

int result = conn.DeleteOrUpdate(query);

// Check update result

if (result > 0)

{

MessageBox.Show("The class data has been updated succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem updating the class information, please check your data!");

return false;

}

}

}

return false;

}

}

}

Equipment.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections;

using System.Windows.Forms;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the EQUIPMENT table.

\* Most closely associated form is frm\_equipment.

\* Most closely associated table is EQUIPMENT.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class Equipment

{

// id\_equipment field from EQUIPMENT table

private int id\_equipment;

public int Id\_equipment

{

get { return id\_equipment; }

set { id\_equipment = value; }

}

// type field from EQUIPMENT table it is an enum field. values: item, set, vehicle

private string type;

public string Type

{

get { return type; }

set { type = value; }

}

// id\_vehicle field from EQUIPMENT table

private int id\_vehicle;

public int Id\_vehicle

{

get { return id\_vehicle; }

set { id\_vehicle = value; }

}

// name field from EQUIPMENT table

private string name;

public string Name

{

get { return name; }

set { name = value; }

}

// description field from EQUIPMENT table

private string description;

public string Description

{

get { return description; }

set { description = value; }

}

// A field with the same name from EQUIPMENT table

private int itemInSet1;

public int ItemInSet1

{

get { return itemInSet1; }

set { itemInSet1 = value; }

}

// A field with the same name from EQUIPMENT table

private int amountInSet1;

public int AmountInSet1

{

get { return amountInSet1; }

set { amountInSet1 = value; }

}

// A field with the same name from EQUIPMENT table

private int itemInSet2;

public int ItemInSet2

{

get { return itemInSet2; }

set { itemInSet2 = value; }

}

// A field with the same name from EQUIPMENT table

private int amountInSet2;

public int AmountInSet2

{

get { return amountInSet2; }

set { amountInSet2 = value; }

}

// A field with the same name from EQUIPMENT table

private int itemInSet3;

public int ItemInSet3

{

get { return itemInSet3; }

set { itemInSet3 = value; }

}

// A field with the same name from EQUIPMENT table

private int amountInSet3;

public int AmountInSet3

{

get { return amountInSet3; }

set { amountInSet3 = value; }

}

// A field with the same name from EQUIPMENT table

private int itemInSet4;

public int ItemInSet4

{

get { return itemInSet4; }

set { itemInSet4 = value; }

}

// A field with the same name from EQUIPMENT table

private int amountInSet4;

public int AmountInSet4

{

get { return amountInSet4; }

set { amountInSet4 = value; }

}

// A field with the same name from EQUIPMENT table

private int itemInSet5;

public int ItemInSet5

{

get { return itemInSet5; }

set { itemInSet5 = value; }

}

// A field with the same name from EQUIPMENT table

private int amountInSet5;

public int AmountInSet5

{

get { return amountInSet5; }

set { amountInSet5 = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_equipment to -1 so the fact of this is a new equipment can be referenced.

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public Equipment()

{

this.id\_equipment = -1;

}

/\*\*

\* @desc Constructor

\* Loads in all fields from a single row of the EQUIPMENT table.

\* @params [int] id\_equipment identifies the equipment uniquely.

\* @return [none] No directly returned data.

\*/

public Equipment(int id\_equipment)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Launch the query to return all fields from a single row of the EQUIPMENT table

List<Hashtable> lhResultset = conn.lhSqlQuery("Select \* from equipment WHERE id\_equipment = '" + id\_equipment + "'");

// Check if we found the equipment

if ((int)lhResultset.Count > 0)

{

// Fill in all equipment fields with table data

this.Id\_equipment = int.Parse(lhResultset[0]["id\_equipment"].ToString());

this.Type = lhResultset[0]["type"].ToString();

this.Id\_vehicle = int.Parse(lhResultset[0]["id\_vehicle"].ToString());

this.Name = lhResultset[0]["name"].ToString();

this.Description = lhResultset[0]["description"].ToString();

// Fill in all equipment set fields with table data

if (this.Type == "set")

{

this.ItemInSet1 = int.Parse(lhResultset[0]["iteminset1"].ToString());

this.ItemInSet2 = int.Parse(lhResultset[0]["iteminset2"].ToString());

this.ItemInSet3 = int.Parse(lhResultset[0]["iteminset3"].ToString());

this.ItemInSet4 = int.Parse(lhResultset[0]["iteminset4"].ToString());

this.ItemInSet5 = int.Parse(lhResultset[0]["iteminset5"].ToString());

this.AmountInSet1 = int.Parse(lhResultset[0]["amountinset1"].ToString());

this.AmountInSet2 = int.Parse(lhResultset[0]["amountinset2"].ToString());

this.AmountInSet3 = int.Parse(lhResultset[0]["amountinset3"].ToString());

this.AmountInSet4 = int.Parse(lhResultset[0]["amountinset4"].ToString());

this.AmountInSet5 = int.Parse(lhResultset[0]["amountinset5"].ToString());

}

}

}

/\*\*

\* @desc Removes the equipment from the EQUIPMENT table.

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem deleting the class.

\*/

public bool RemoveEquipment()

{

if (this.Id\_equipment != -1)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create the delete query

string query = "DELETE FROM equipment WHERE id\_equipment = '" + this.Id\_equipment + "'";

// Launch delete query

int result = conn.DeleteOrUpdate(query);

// Check deletion result

if (result > 0)

{

MessageBox.Show("The equipment data has been deleted succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem deleting the equipment!");

return false;

}

}

return false;

}

/\*\*

\* @desc This method will save or update an equipment in the EQUIPMENT table

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem saving/updating the equipment

\*/

public bool SaveEquipment()

{

string saveEquipmentQuery;

// Checking user input

if (this.Name == "")

{

MessageBox.Show("Please Insert a name.");

}

else

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Check whether there is a new id\_equipment assigned to this equipment,

// if not then this a new equipment to save

if (this.Id\_equipment == -1)

{

// Create the save query

saveEquipmentQuery = "insert into `gym`.`equipment` (`id\_equipment`, `type`, `id\_vehicle`, `name`, `description`, `iteminset1`, `iteminset2`, `iteminset3`, `iteminset4`, `iteminset5`, `amountinset1`, `amountinset2`, `amountinset3`, `amountinset4`, `amountinset5`) values " +

"(NULL, '" + this.Type + "', '" + this.Id\_vehicle + "', '" + this.Name + "', '" + this.Description

+ "', '" + this.ItemInSet1 + "', '" + this.ItemInSet2 + "', '" + this.ItemInSet3 + "', '" + this.ItemInSet4 + "', '" + this.ItemInSet5

+ "', '" + this.AmountInSet1 + "', '" + this.AmountInSet2 + "', '" + this.AmountInSet3 + "', '" + this.AmountInSet4 + "', '" + this.AmountInSet5 + "')";

// Launch save query

int id\_equipment = conn.InsertToDB(saveEquipmentQuery);

// Check saving result

if (id\_equipment != -1)

{

this.Id\_equipment = id\_equipment;

MessageBox.Show("The new equipment has been added to the databse succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem adding the new equipment, please check your data!");

return false;

}

}

// If an id\_equipment already exists for this equipment, then this is an existing equipment to update

else

{

// Create update query

string updateEquimentQuery = "UPDATE equipment SET type = '" + this.Type

+ "', id\_vehicle = '" + this.Id\_vehicle

+ "', name = '" + this.Name

+ "', description = '" + this.Description

+ "', iteminset1 = '" + this.ItemInSet1

+ "', iteminset2 = '" + this.ItemInSet2

+ "', iteminset3 = '" + this.ItemInSet3

+ "', iteminset4 = '" + this.ItemInSet4

+ "', iteminset5 = '" + this.ItemInSet5

+ "', amountinset1 = '" + this.AmountInSet1

+ "', amountinset2 = '" + this.AmountInSet2

+ "', amountinset3 = '" + this.AmountInSet3

+ "', amountinset4 = '" + this.AmountInSet4

+ "', amountinset5 = '" + this.AmountInSet5 + "' "

+ " WHERE id\_equipment = '" + this.Id\_equipment + "'";

// Launch update query

int result = conn.DeleteOrUpdate(updateEquimentQuery);

// Check update result

if (result > 0)

{

MessageBox.Show("The equipment data has been updated succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem updating the equipment information, please check your data!");

return false;

}

}

}

return false;

}

}

}

EquipmentBooked.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections;

using System.Windows.Forms;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the EQUIPMENT\_BOOKINGS table.

\* Most closely associated forms are frm\_member, frm\_staff, frm\_class.

\* Most closely associated table is EQUIPMENT\_BOOKINGS.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class EquipmentBooked

{

// A field with the same name from EQUIPMENT\_BOOKINGS table

private int id\_eq\_booking;

public int Id\_eq\_booking

{

get { return id\_eq\_booking; }

set { id\_eq\_booking = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private string id\_member;

public string Id\_member

{

get { return id\_member; }

set { id\_member = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private string id\_staff;

public string Id\_staff

{

get { return id\_staff; }

set { id\_staff = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private string id\_class\_instance;

public string Id\_class\_instance

{

get { return id\_class\_instance; }

set { id\_class\_instance = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private string dateStart;

public string DateStart

{

get { return dateStart; }

set { dateStart = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private string dateDue;

public string DateDue

{

get { return dateDue; }

set { dateDue = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private int id\_equipment;

public int Id\_equipment

{

get { return id\_equipment; }

set { id\_equipment = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private int borrowedAmount;

public int BorrowedAmount

{

get { return borrowedAmount; }

set { borrowedAmount = value; }

}

// A field with the same name from EQUIPMENT\_BOOKINGS table

private bool isReturned;

public bool IsReturned

{

get { return isReturned; }

set { isReturned = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_eq\_booking to -1 so the fact of this is a new equipment booking can be referenced.

\*

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public EquipmentBooked()

{

this.id\_eq\_booking = -1;

}

/\*\*

\* @desc Constructor

\* Loads in the id\_eq\_booking into its corresponding field to open up a placeholder to

\* fill with data to modify an existing booking

\* @params [int] id\_class identifies the class uniquely.

\* @return [none] No directly returned data.

\*/

public EquipmentBooked(int id\_eq\_booking)

{

this.Id\_eq\_booking = id\_eq\_booking;

}

/\*\*

\* @desc This method will save or update an equipment booking in the EQUIPMENT\_BOOKINGS table

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem saving/updating the equipment

\*/

public bool SaveEquipmentBooking()

{

string query;

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Check whether there is a new id\_eq\_booking assigned to this booking,

// if not then this a new equipment booking to save

if (this.Id\_eq\_booking == -1)

{

// Create the save query

query = "insert into `gym`.`equipment\_bookings` (`id\_eq\_booking`, `id\_staff`, `id\_member`, `id\_class\_instance`, `date\_start`, `date\_due`, `id\_equipment`, `borrowedamount`,`isreturned`) values " +

"(NULL, " + this.Id\_staff + ", " + this.Id\_member + ", " + this.Id\_class\_instance + ", '" + this.DateStart + "', '" + this.DateDue

+ "', " + this.Id\_equipment + ", " + this.BorrowedAmount + ", NULL)";

// Launch save query

int id\_eq\_booking = conn.InsertToDB(query);

// Check saving result

if (id\_eq\_booking != -1)

{

this.Id\_eq\_booking = id\_eq\_booking;

MessageBox.Show("The new equipment booking has been added to the databse succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem adding the new equipment booking, please check your data!");

return false;

}

}

// If an id\_eq\_booking already exists for this booking, then this is an existing booking to update

else

{

// Create update query

query = "UPDATE `gym`.`equipment\_bookings` SET `borrowedamount` = " + this.BorrowedAmount + ", `isreturned`= " + this.IsReturned + " WHERE id\_eq\_booking = '" + this.Id\_eq\_booking + "'";

// Launch update query

int result = conn.DeleteOrUpdate(query);

// Check update result

if (result > 0)

{

MessageBox.Show("The equipment booking data has been updated succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem updating the equipment booking information, please check your data!");

return false;

}

}

//return true;

}

}

}

Member.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Collections;

using System.Drawing;

using MySql.Data.MySqlClient;

using System.Data.SqlClient;

using System.Data;

using System.IO;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the MEMBERS table.

\* Most closely associated form is frm\_member.

\* Most closely associated table is MEMBERS.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class Member : Person

{

// A field with the same name from MEMBERS table

private int id\_member;

public int Id\_member

{

get { return id\_member; }

set { id\_member = value; }

}

// Field member\_number from MEMBERS table

private string memberNumber;

public string MemberNumber

{

get { return memberNumber; }

set { memberNumber = value; }

}

// Field type from MEMBERS table

private string type;

public string Type

{

get { return type; }

set { type = value; }

}

// Field is\_active from MEMBERS table

private bool isActive;

public bool IsActive

{

get { return isActive; }

set { isActive = value; }

}

// Field emerg\_contact\_name from MEMBERS table

private string emergContactName;

public string EmergContactName

{

get { return emergContactName; }

set { emergContactName = value; }

}

// Field emerg\_contact\_relation from MEMBERS table

private string emergContactRelation;

public string EmergContactRelation

{

get { return emergContactRelation; }

set { emergContactRelation = value; }

}

// Field emerg\_contact\_phone from MEMBERS table

private string emergContactPhone;

public string EmergContactPhone

{

get { return emergContactPhone; }

set { emergContactPhone = value; }

}

// Field emerg\_contact\_mobile from MEMBERS table

private string emergContactMobile;

public string EmergContactMobile

{

get { return emergContactMobile; }

set { emergContactMobile = value; }

}

// Field medical\_allergies from MEMBERS table

private string medicalAllergies;

public string MedicalAllergies

{

get { return medicalAllergies; }

set { medicalAllergies = value; }

}

// Field medical\_notes from MEMBERS table

private string medicalNotes;

public string MedicalNotes

{

get { return medicalNotes; }

set { medicalNotes = value; }

}

// Field id\_file from MEMBERS and FILE tabe for referencing from FILE table

private string id\_file;

public string Id\_file

{

get { return id\_file; }

set { id\_file = value; }

}

// Field file\_name from FILE table

private string fileName;

public string FileName

{

get { return fileName; }

set { fileName = value; }

}

// Variable for storing file path

private string filePath;

public string FilePath

{

get { return filePath; }

set { filePath = value; }

}

// Field medical\_doctor\_name from MEMBERS table

private string medicalDoctorName;

public string MedicalDoctorName

{

get { return medicalDoctorName; }

set { medicalDoctorName = value; }

}

// Field medical\_phone from MEMBERS table

private string medicalPhone;

public string MedicalPhone

{

get { return medicalPhone; }

set { medicalPhone = value; }

}

// Field ismale from MEMBERS table

private string gender;

public string Gender

{

get { return gender; }

set { gender = value; }

}

// a User (User.cs) object is stored here

private User clUser;

internal User ClUser

{

get { return clUser; }

set { clUser = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_member to -1 so the fact of this is a new member can be referenced.

\* Creates a new user parent class instance.

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public Member()

{

this.id\_member = -1;

this.clUser = new User();

}

/\*\*

\* @desc Constructor

\* Loads in all fields from a single row of the MEMBERS table.

\* @params [int] id\_member identifies the member uniquely.

\* @return [none] No directly returned data.

\*/

public Member(int id\_member)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Launch the query to return all fields from a single row of the MEMBERS table

List<Hashtable> lhResultset = conn.lhSqlQuery("Select \* from members m, users u where u.id\_user = m.id\_user AND m.id\_member = '" + id\_member + "'");

// Check if we found the member

if ((int)lhResultset.Count > 0)

{

// Fill in all member and parent user fields with table data

this.clUser = new User();

this.clUser.Id\_user = int.Parse(lhResultset[0]["id\_user"].ToString());

this.clUser.Login = lhResultset[0]["login"].ToString();

this.clUser.Password = lhResultset[0]["password"].ToString();

this.clUser.Profile = lhResultset[0]["profile"].ToString();

this.IsActive = (lhResultset[0]["is\_active"].ToString() == "True") ? true : false;

this.Id\_member = int.Parse(lhResultset[0]["id\_member"].ToString());

this.Address\_2 = lhResultset[0]["address\_2"].ToString();

this.Address\_1 = lhResultset[0]["address\_1"].ToString();

this.Birthdate = lhResultset[0]["birthdate"].ToString();

this.City = lhResultset[0]["city"].ToString();

this.County = lhResultset[0]["county"].ToString();

this.Email = lhResultset[0]["email"].ToString();

this.EmergContactMobile = lhResultset[0]["emerg\_contact\_mobile"].ToString();

this.EmergContactName = lhResultset[0]["emerg\_contact\_name"].ToString();

this.EmergContactPhone = lhResultset[0]["emerg\_contact\_phone"].ToString();

this.EmergContactRelation = lhResultset[0]["emerg\_contact\_relation"].ToString();

this.FirstName = lhResultset[0]["firstName"].ToString();

this.LastName = lhResultset[0]["lastName"].ToString();

this.MedicalAllergies = lhResultset[0]["medical\_allergies"].ToString();

this.MedicalDoctorName = lhResultset[0]["medical\_doctor\_name"].ToString();

this.MedicalNotes = lhResultset[0]["medical\_notes"].ToString();

this.MedicalPhone = lhResultset[0]["medical\_phone"].ToString();

this.MemberNumber = lhResultset[0]["member\_number"].ToString();

this.Id\_file = lhResultset[0]["id\_file"].ToString();

this.PostalCode = lhResultset[0]["postalcode"].ToString();

this.Type = lhResultset[0]["type"].ToString();

this.Mobile = lhResultset[0]["mobile"].ToString();

this.Phone = lhResultset[0]["phone"].ToString();

this.Gender = lhResultset[0]["gender"].ToString();

}

}

/\*\*

\* @desc Creates a new payment for the opened member

\* @params [decimal] amount: the amount payed

\* @params [string] date: the date of the payment

\* @params [string] desc: description/comments

\* @params [string] receiptNumber: receipt number

\* @params [string] paymentMethod: card/cash/cheque/bank transfer

\* @params [string] receivedBy: name of person, who received the payment

\* @return [bool] Returns true in case of success, false if there was a problem

\*/

public bool AddPayment(Decimal amount, string date, string desc, string receiptNumber, string paymentMethod, string receivedBy)

{

// The payment can be added to existing members only, not new members

if (Id\_member != -1)

{

// Create a payment object and copy into all payment data

Payment clPayment = new Payment();

clPayment.Amount = amount;

clPayment.Date = date;

clPayment.ClMember = this;

clPayment.Details = desc;

clPayment.ReceiptNumber = receiptNumber;

clPayment.PaymentMethod = paymentMethod;

clPayment.ReceivedBy = receivedBy;

// Save payment

if (clPayment.SavePayment())

return true;

else

return false;

}

else

return false;

}

/\*\*

\* @desc This method will save the object into the database

\* @return [bool] Returns true in case of success, false if there was a problem

\*/

public bool SaveMember()

{

// Convert date into mysql format

string mysqlDate = Utils.sGetMysqlDate(this.Birthdate);

string query;

// Check Birthdate format

if (mysqlDate == "0000-00-00")

{

MessageBox.Show("The Date of Birth is in incorrect format!");

}

// Check e-mail format

else if (Utils.bValidateEmail(this.Email) == false)

{

MessageBox.Show("The E-Mail address is incorrect!");

}

else

{

// First the user object is filled

clUser.IsActive = (this.IsActive) ? true : false;

clUser.Login = this.Email;

clUser.Password = mysqlDate;

clUser.Profile = "member";

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// If the User details were correctly saved

if (clUser.SaveUser())

{

// Check if there is a new picture to save

if ((this.FilePath != null) && (this.FilePath.Length > 1))

{

this.Id\_file = conn.uploadFileToDB(this.FilePath, this.FileName);

}

// The insert query is launched in case of existing members only, not new members

if (this.Id\_member == -1)

{

// Create insert query

query = "insert into `gym`.`members` (`id\_member`, `firstName`, `lastName`, `birthdate`, `address\_1`, `city`, `county`, `postalcode`, `type`, `id\_user`, `is\_active`, `address\_2`, `emerg\_contact\_name`, `emerg\_contact\_relation`, `emerg\_contact\_phone`, `emerg\_contact\_mobile`, `medical\_allergies`, `medical\_notes`, `id\_file`, `medical\_doctor\_name`, `medical\_phone`, `email`, `member\_number`, `phone`,`mobile`,`gender`) values " +

"(NULL, '" + this.FirstName + "', '" + this.LastName + "', '" + mysqlDate + "', '" + this.Address\_1 + "', '" + this.City + "', '" + this.County + "', '" + this.PostalCode + "', '" + this.Type + "', '" + clUser.Id\_user + "', '" + ((this.IsActive) ? "1" : "0") + "', '" + this.Address\_2 + "', '" + this.EmergContactName + "', '" + this.EmergContactRelation + "', '" + this.EmergContactPhone + "', '" + this.EmergContactMobile + "', '" + this.MedicalAllergies + "', '" + this.MedicalNotes + "', '" + this.Id\_file + "', '" + this.MedicalDoctorName + "', '" + this.MedicalPhone + "', '" + this.Email + "', '" + this.MemberNumber + "','" + this.Phone + "','" + this.Mobile + "','" + this.Gender + "')";

// Launch insert query

int id\_member = conn.InsertToDB(query);

// Check if the insert was successful

if (id\_member != -1)

{

this.Id\_member = id\_member;

MessageBox.Show("The new member has been added to the databse succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem adding the new user, please check your data!");

clUser.DeleteUser();

return false;

}

}

// This is a member update

else

{

// Create update query

query = "UPDATE members SET firstName = '" + this.FirstName + "', lastName = '" + this.LastName + "', birthdate = '" + mysqlDate + "', address\_1 = '" + this.Address\_1 + "', city = '" + this.City + "', county = '" + this.County + "', postalcode = '" + this.PostalCode + "', type = '" + this.Type + "', is\_active = " + ((this.IsActive) ? "1" : "0") + ", address\_2 = '" + this.Address\_2 + "', emerg\_contact\_name = '" + this.EmergContactName + "', emerg\_contact\_relation = '" + this.EmergContactRelation + "', emerg\_contact\_phone = '" + this.EmergContactPhone + "', emerg\_contact\_mobile = '" + this.EmergContactMobile + "', medical\_allergies = '" + this.MedicalAllergies + "', medical\_notes = '" + this.MedicalNotes + "', id\_file = '" + this.Id\_file + "', medical\_doctor\_name = '" + this.MedicalDoctorName + "', medical\_phone = '" + this.MedicalPhone + "', email = '" + this.Email + "', phone = '" + this.Phone + "', mobile = '" + this.Mobile +

"', gender = '" + this.Gender + "' " + " WHERE id\_member = '"+this.Id\_member+"'";

// Launch update query

int result = conn.DeleteOrUpdate(query);

// Check if the update was successful

if (result > 0)

{

MessageBox.Show("The member data has been updated succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem updating the user information, please check your data!");

clUser.DeleteUser();

return false;

}

}

}

// If the user saving was false, then it was becuase of duplicate e-mail at this point

else

{

MessageBox.Show("The e-mail already exists in the database! Please choose another one.");

return false;

}

}

return false;

}

/\*\*

\* @desc This method will set a member to be inactive in the database

\* @return [bool] Returns true in case of success, false if there was a problem

\*/

public bool RemoveMember()

{

// If an existing member is currently loaded in

if (this.Id\_member != -1)

{

// Set his active status to inactive

this.IsActive = false;

return this.SaveMember();

}

return false;

}

}

}

Payment.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Gym\_administration

{

/\*\*

\* @desc It temporarily holds payment details for saving a new payment

\* Most closely associated form is frm\_member.

\* Most closely associated table is MEMBERS.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class Payment

{

// A field with the same name from PAYMENTS table

private Decimal amount;

public Decimal Amount

{

get { return amount; }

set { amount = value; }

}

// A field with the same name from PAYMENTS table

private string date;

public string Date

{

get { return date; }

set { date = value; }

}

// A field with the same name from PAYMENTS table

private string details;

public string Details

{

get { return details; }

set { details = value; }

}

// A Member (Member.cs) object is stored here

private Member clMember;

internal Member ClMember

{

get { return clMember; }

set { clMember = value; }

}

// A field with the same name from PAYMENTS table

private string paymentMethod;

public string PaymentMethod

{

get { return paymentMethod; }

set { paymentMethod = value; }

}

// A field with the same name from PAYMENTS table

private string receiptNumber;

public string ReceiptNumber

{

get { return receiptNumber; }

set { receiptNumber = value; }

}

// A field with the same name from PAYMENTS table

private string receivedBy;

public string ReceivedBy

{

get { return receivedBy; }

set { receivedBy = value; }

}

/\*\*

\* @desc Default constructor.

\* It creates a payment object for holding and saving payment data

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public Payment()

{

// Nothing here currently.

}

/\*\*

\* @desc This method will save the object into the database

\* @return [bool] Returns true in case of success, false if there was a problem

\*/

public bool SavePayment()

{

// If this is an existing member's payment

if (this.ClMember.Id\_member != -1)

{

// Create insert query

string query = "insert into `gym`.`payments` (`id\_payment`, `id\_member`, `date`, `amount`, `details`,`receiptnumber`,`paymentmethod`,`receivedby`) values (NULL, '" + this.ClMember.Id\_member + "', '" + this.Date + "', '" + this.Amount + "', '" + this.Details + "', '" + this.ReceiptNumber + "', '" + this.PaymentMethod + "', '" + this.ReceivedBy + "');";

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Launch insert query

int payment = conn.InsertToDB(query);

// Check if the insert was succesful

if (payment != -1)

return true;

}

return false;

}

//Modifying payments? Currently NO!

//this.PaymentMethod = lhResultset[0]["payment\_method"].ToString();

//payment\_method = '"+this.PaymentMethod+"'

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Gym\_administration

{

/\*\*

\* @desc It's an abstract class because it is not instantiable

\* @params [none]

\* @return [none]

\*/

public abstract class Person

{

// Everything here corresponds with fields from STAFF and MEMBER tables

private string firstName;

public string FirstName

{

get { return firstName; }

set { firstName = value; }

}

private string lastName;

public string LastName

{

get { return lastName; }

set { lastName = value; }

}

private string birthdate;

public string Birthdate

{

get { return birthdate; }

set { birthdate = value; }

}

private string address\_1;

public string Address\_1

{

get { return address\_1; }

set { address\_1 = value; }

}

private string address\_2;

public string Address\_2

{

get { return address\_2; }

set { address\_2 = value; }

}

private string city;

public string City

{

get { return city; }

set { city = value; }

}

private string county;

public string County

{

get { return county; }

set { county = value; }

}

private string postalCode;

public string PostalCode

{

get { return postalCode; }

set { postalCode = value; }

}

private string email;

public string Email

{

get { return email; }

set { email = value; }

}

private string phone;

public string Phone

{

get { return phone; }

set { phone = value; }

}

private string mobile;

public string Mobile

{

get { return mobile; }

set { mobile = value; }

}

}

}

Room.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections;

using System.Windows.Forms;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the ROOMS table.

\* Most closely associated form is frm\_room.

\* Most closely associated table is ROOMS.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class Room

{

// A field with the same name from MEMBERS table

private int id\_room;

public int Id\_room

{

get { return id\_room; }

set { id\_room = value; }

}

// A field with the same name from MEMBERS table

private string name;

public string Name

{

get { return name; }

set { name = value; }

}

// A field with the same name from MEMBERS table

private int size;

public int Size

{

get { return size; }

set { size = value; }

}

// A field with the same name from MEMBERS table

private string description;

public string Description

{

get { return description; }

set { description = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_room to -1 so the fact of this is a new room can be referenced.

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public Room()

{

this.id\_room = -1;

}

/\*\*

\* @desc Constructor

\* Loads in all fields from a single row of the ROOMS table.

\* @params [int] id\_room identifies the room uniquely.

\* @return [none] No directly returned data.

\*/

public Room(int id\_room)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Launch the query to return all fields from a single row of the ROOMS table

List<Hashtable> lhResultset = conn.lhSqlQuery("Select \* from rooms WHERE id\_room = '" + id\_room + "'");

// Check if we found the room

if ((int)lhResultset.Count > 0)

{

// Fill in all room fields with table data

this.Id\_room = int.Parse(lhResultset[0]["id\_room"].ToString());

this.Size = int.Parse(lhResultset[0]["size"].ToString());

this.Description = lhResultset[0]["description"].ToString();

this.Name = lhResultset[0]["name"].ToString();

}

}

/\*\*

\* @desc Removes the room from the ROOMS table.

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem deleting the class.

\*/

public bool RemoveRoom()

{

if (this.Id\_room != -1)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create the delete query

string query = "DELETE FROM rooms WHERE id\_room = '"+this.Id\_room+"'";

// Launch delete query

int result = conn.DeleteOrUpdate(query);

// Check deletion result

if (result > 0)

{

MessageBox.Show("The room data has been deleted succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem deleting the room!");

return false;

}

}

return false;

}

/\*\*

\* @desc This method will save or update a room in the ROOMS table

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem saving/updating the class

\*/

public bool SaveRoom()

{

string query;

// Checking user input

if (this.Name == "")

{

MessageBox.Show("Please Insert a name.");

}

else

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Check whether there is a new id\_room assigned to this room,

// if not then this a new room to save

if (this.Id\_room == -1)

{

// Create insert query

query = "insert into `gym`.`rooms` (`id\_room`, `name`, `size`, `description`) values " +

"(NULL, '" + this.Name + "', '" + this.Size + "', '" + this.Description + "')";

// Launch insert query

int id\_room = conn.InsertToDB(query);

// Check saving result

if (id\_room != -1)

{

this.Id\_room = id\_room;

MessageBox.Show("The new room has been added to the databse succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem adding the new room, please check your data!");

return false;

}

}

// If an id\_room already exists for this room instance, then this is an existing room to update

else

{

// Create update query

query = "UPDATE rooms SET name = '" + this.Name + "', size = '" + this.Size + "', description = '" + this.Description + "' " +

" WHERE id\_room = '" + this.Id\_room + "'";

// Launch update query

int result = conn.DeleteOrUpdate(query);

// Check update reults

if (result > -1)

{

MessageBox.Show("The room data has been updated succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem updating the room information, please check your data!");

return false;

}

}

}

return false;

}

}

}

Staff.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Collections;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the STAFF table.

\* Most closely associated form is frm\_staff.

\* Most closely associated table is STAFF.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class Staff : Person

{

// A field from the STAFF table

private int id\_staff;

public int Id\_staff

{

get { return id\_staff; }

set { id\_staff = value; }

}

// A field from the STAFF table

private string position;

public string Position

{

get { return position; }

set { position = value; }

}

// A field from the STAFF table

private string contractType;

public string ContractType

{

get { return contractType; }

set { contractType = value; }

}

// A field from the STAFF table

private string emergContactName;

public string EmergContactName

{

get { return emergContactName; }

set { emergContactName = value; }

}

// A field from the STAFF table

private string emergContactRelation;

public string EmergContactRelation

{

get { return emergContactRelation; }

set { emergContactRelation = value; }

}

// A field from the STAFF table

private string emergContactPhone;

public string EmergContactPhone

{

get { return emergContactPhone; }

set { emergContactPhone = value; }

}

// A field from the STAFF table

private string emergContactMobile;

public string EmergContactMobile

{

get { return emergContactMobile; }

set { emergContactMobile = value; }

}

// A field from the STAFF table

private string medicalAllergies;

public string MedicalAllergies

{

get { return medicalAllergies; }

set { medicalAllergies = value; }

}

// A field from the STAFF table

private string medicalNotes;

public string MedicalNotes

{

get { return medicalNotes; }

set { medicalNotes = value; }

}

// A field from the STAFF table

private string qualifications;

public string Qualifications

{

get { return qualifications; }

set { qualifications = value; }

}

// A field from the STAFF table

private string picture;

public string Picture

{

get { return picture; }

set { picture = value; }

}

// A field from the STAFF table

private string medicalDoctorName;

public string MedicalDoctorName

{

get { return medicalDoctorName; }

set { medicalDoctorName = value; }

}

// A field from the STAFF table

private string medicalPhone;

public string MedicalPhone

{

get { return medicalPhone; }

set { medicalPhone = value; }

}

// a User (User.cs) object is stored here

private User clUser;

internal User ClUser

{

get { return clUser; }

set { clUser = value; }

}

// A field from the STAFF table

private string natInsNumb;

public string NatInsNumb

{

get { return natInsNumb; }

set { natInsNumb = value; }

}

// A field from the STAFF table

private string sContractStart;

public string SContractStart

{

get { return sContractStart; }

set { sContractStart = value; }

}

// A field from the STAFF table

private string sContractFinish;

public string SContractFinish

{

get { return sContractFinish; }

set { sContractFinish = value; }

}

// Field is\_active from STAFF table

private bool isActive;

public bool IsActive

{

get { return isActive; }

set { isActive = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_staff to -1 so the fact of this is a new staff can be referenced.

\* Creates a new user parent class instance.

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public Staff()

{

this.Id\_staff = -1;

this.clUser = new User();

}

/\*\*

\* @desc Constructor

\* Loads in all fields from a single row of the Staff table.

\* @params [int] id\_staff identifies the staff uniquely.

\* @return [none] No directly returned data.

\*/

public Staff(int id\_staff)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Launch the query to return all fields from a single row of the STAFF table

List<Hashtable> lhResultset = conn.lhSqlQuery("Select \* from staff s, users u where u.id\_user = s.id\_user AND s.id\_staff = '" + id\_staff + "'");

// Check if we found the staff

if ((int)lhResultset.Count > 0)

{

// Fill in all staff and parent user fields with table data

this.clUser = new User();

this.clUser.Id\_user = int.Parse(lhResultset[0]["id\_user"].ToString());

this.clUser.Login = lhResultset[0]["login"].ToString();

this.clUser.Password = lhResultset[0]["password"].ToString();

this.clUser.Profile = lhResultset[0]["profile"].ToString();

this.Id\_staff = int.Parse(lhResultset[0]["id\_staff"].ToString());

this.Address\_2 = lhResultset[0]["address\_2"].ToString();

this.Address\_1 = lhResultset[0]["address\_1"].ToString();

this.Birthdate = lhResultset[0]["birthdate"].ToString();

this.City = lhResultset[0]["city"].ToString();

this.IsActive = (lhResultset[0]["is\_active"].ToString() == "True") ? true : false;

this.County = lhResultset[0]["county"].ToString();

this.Email = lhResultset[0]["email"].ToString();

this.EmergContactName = lhResultset[0]["emerg\_contact\_name"].ToString();

this.EmergContactPhone = lhResultset[0]["emerg\_contact\_telephone"].ToString();

this.EmergContactRelation = lhResultset[0]["emerg\_contact\_relation"].ToString();

this.EmergContactMobile = lhResultset[0]["emerg\_contact\_mobile"].ToString();

this.FirstName = lhResultset[0]["firstName"].ToString();

this.LastName = lhResultset[0]["lastName"].ToString();

this.MedicalAllergies = lhResultset[0]["allergies"].ToString();

this.MedicalDoctorName = lhResultset[0]["medical\_doctor\_name"].ToString();

this.MedicalNotes = lhResultset[0]["medicalNotes"].ToString();

this.Qualifications = lhResultset[0]["qualifications"].ToString();

this.MedicalPhone = lhResultset[0]["medical\_phone"].ToString();

this.Picture = "none";

this.PostalCode = lhResultset[0]["postalcode"].ToString();

this.ContractType = lhResultset[0]["contract\_type"].ToString();

this.Mobile = lhResultset[0]["mobile"].ToString();

this.Phone = lhResultset[0]["phone"].ToString();

this.Position = lhResultset[0]["position"].ToString();

this.NatInsNumb = lhResultset[0]["natinsnumber"].ToString();

this.SContractFinish = lhResultset[0]["contract\_finish"].ToString();

this.SContractStart = lhResultset[0]["contract\_start"].ToString();

}

}

/\*\*

\* @desc This method will save the object into the database

\* @return [bool] Returns true in case of success, false if there was a problem

\*/

public bool SaveStaff()

{

string query;

// Convert dates into mysql format

string sMysqlDate = Utils.sGetMysqlDate(this.Birthdate);

string sMysqlStartDate = Utils.sGetMysqlDate(this.SContractStart);

string sMysqlFinishDate = Utils.sGetMysqlDate(this.SContractFinish);

// Check user input format

if (sMysqlDate == "0000-00-00")

MessageBox.Show("The Date of Birth is in incorrect format!");

else if (sMysqlStartDate == "0000-00-00")

MessageBox.Show("The Contract Start date is in incorrect format!");

else if (sMysqlFinishDate == "0000-00-00")

MessageBox.Show("The Contract Finish date is in incorrect format!");

else if (Utils.bValidateEmail(this.Email) == false)

MessageBox.Show("The E-Mail address is incorrect");

else

{

// First the user object is filled

clUser.Login = this.Email;

clUser.Password = this.Birthdate;

clUser.Profile = "staff";

clUser.IsActive = true;

if (clUser.SaveUser())

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// If this is a new staff

if (Id\_staff == -1)

{

// Create insert query

query = "insert into `gym`.`staff` (`id\_staff`, `firstName`, `lastName`, `birthdate`, `address\_1`, `city`, `county`, `postalcode`, `contract\_type`, `natinsnumber`, `position`, `contract\_start`, `contract\_finish`, `id\_user`, `address\_2`, `emerg\_contact\_name`, `emerg\_contact\_telephone`, `emerg\_contact\_relation`, `allergies`, `medicalNotes`, `qualifications`, `phone`, `mobile`, `email`, `emerg\_contact\_mobile`, `medical\_doctor\_name`, `medical\_phone`, `is\_active`) values " +

"(NULL, '" + this.FirstName + "', '" + this.LastName + "', '" + sMysqlDate + "', '" + this.Address\_1 + "', '" + this.City + "', '" + this.County + "', '" + this.PostalCode + "', '" + this.ContractType + "', '" + this.NatInsNumb + "', '" + this.Position + "', '" + sMysqlStartDate + "', '" + sMysqlFinishDate + "', '" + clUser.Id\_user + "', '" + this.Address\_2 + "', '" + this.EmergContactName + "', '" + this.EmergContactPhone + "', '" + this.EmergContactRelation + "', '" + this.MedicalAllergies + "', '" + this.MedicalNotes + "','" + this.Qualifications + "', '" + this.Phone + "','" + this.Mobile + "','" + this.Email + "','" + this.EmergContactMobile + "','" + this.MedicalDoctorName + "','" + this.MedicalPhone + "','" + ((this.IsActive) ? "1" : "0") +"')";

// Launch insert query

int id\_staff = conn.InsertToDB(query);

// Check if the insert was successful

if (id\_staff != -1)

{

Id\_staff = id\_staff;

MessageBox.Show("The new staff member has been added to the databse succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem adding the new user, please check your data!");

clUser.DeleteUser();

return false;

}

}

// This is a staff update

else

{

// Create update query

query = "UPDATE staff SET firstName = '" + this.FirstName + "', lastName = '" + this.LastName + "', birthdate = '" + sMysqlDate + "', address\_1 = '" + this.Address\_1 + "', city = '" + this.City + "', county = '" + this.County + "', postalcode = '" + this.PostalCode + "', contract\_type = '" + this.ContractType + "', contract\_start = '" + sMysqlStartDate + "', contract\_finish = '" + sMysqlFinishDate + "', address\_2 = '" + this.Address\_2 + "', emerg\_contact\_name = '" + this.EmergContactName + "', emerg\_contact\_relation = '" + this.EmergContactRelation + "', emerg\_contact\_telephone = '" + this.EmergContactPhone + "', emerg\_contact\_mobile = '" + this.EmergContactMobile + "', allergies = '" + this.MedicalAllergies + "', medicalNotes = '" + this.MedicalNotes + "', qualifications = '" + this.Qualifications + "', medical\_doctor\_name = '" + this.MedicalDoctorName + "', medical\_phone = '" + this.MedicalPhone + "', email = '" + this.Email + "', phone = '" + this.Phone + "', mobile = '" + this.Mobile + "', natinsnumber = '" + this.NatInsNumb + "', position = '" + this.Position + "', is\_active = " + ((this.IsActive) ? "1" : "0") +

" WHERE id\_staff = '"+this.id\_staff+"'";

// Launch update query

int result = conn.DeleteOrUpdate(query);

// Check if the update was successful

if (result > 0)

{

MessageBox.Show("The staff data has been updated succesfully!");

return true;

}

else

{

MessageBox.Show("There was a problem updating the user information, please check your data!");

clUser.DeleteUser();

return false;

}

}

}

// If the user saving was false, then it was becuase of duplicate e-mail at this point

else

{

MessageBox.Show("The e-mail already exists in the database!, please choose another one.");

return false;

}

}

return false;

}

/\*\*

\* @desc This method will set a staff to be inactive in the database

\* @return [bool] Returns true in case of success, false if there was a problem

\*/

public bool RemoveStaff()

{

// If an existing member is currently loaded in

if (this.Id\_staff != -1)

{

// Set his active status to inactive

this.IsActive = false;

return this.SaveStaff();

}

return false;

}

}

}

User.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Collections;

namespace Gym\_administration

{

/\*\*

\* @desc It holds data and modifying methods for the USERS table.

\* Most closely associated forms are frm\_member and frm\_staff.

\* Most closely associated table is MEMBERS and STAFF.

\* @params [none] Incoming parameters are described at the individual constructors.

\* @return [none] No directly returned data.

\* Returns of public methods are described at the individual methods.

\*/

class User

{

// A field with the same name from USERS table

private int id\_user;

public int Id\_user

{

get { return id\_user; }

set { id\_user = value; }

}

// A field with the same name from USERS table

private string login;

public string Login

{

get { return login; }

set { login = value; }

}

// A field with the same name from USERS table

private string password;

public string Password

{

get { return password; }

set { password = value; }

}

// A field with the same name from USERS table

private string profile;

public string Profile

{

get { return profile; }

set { profile = value; }

}

// A field with the same name from USERS table

private bool isActive;

public bool IsActive

{

get { return isActive; }

set { isActive = value; }

}

/\*\*

\* @desc Default constructor.

\* Sets id\_class to -1 so the fact of this is a new user can be referenced.

\* @params [none] No input parameter.

\* @return [none] No directly returned data.

\*/

public User()

{

this.Id\_user = -1;

}

/\*\*

\* @desc Constructor

\* Loads in all fields from a single row of the USERS table.

\* @params [int] id\_user identifies the user uniquely.

\* @return [none] No directly returned data.

\*/

public User(int id\_user)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create query

string query = "SELECT \* FROM users WHERE id\_user = '"+id\_user.ToString()+"'";

// Launch the query to return all fields from a single row of the USERS table

List<Hashtable> lhResultset = conn.lhSqlQuery(query);

// Check the user was found

if ((int)lhResultset.Count > 0)

{

// Fill in all user fields with table data

this.Id\_user = int.Parse(lhResultset[0]["id\_user"].ToString());

this.IsActive = true;

this.Login = lhResultset[0]["login"].ToString();

this.Password = lhResultset[0]["password"].ToString();

this.Profile = lhResultset[0]["profile"].ToString();

}

else

MessageBox.Show("The User could not be found!");

}

/\*\*

\* @desc This method will update the password for the user in the USERS table

\* @params [int] id\_user identifies the user uniquely.

\* @params [string] oldPassword: Old password of the user

\* @params [string] newPassword: The new password of the user

\* @return [bool] Returns true in case of success, false if there was problem updating the password

\*/

public bool UpdatePassword(int id\_User, string oldPassword, string newPassword)

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create update password query

string query = "UPDATE users SET password = MD5('" + newPassword + "') WHERE id\_user = '" + id\_user + "' AND password = MD5('" + oldPassword + "')";

// Launch update password query

int result = conn.DeleteOrUpdate(query);

// Check update result

if (result > 0)

return true;

else

return false;

}

/\*\*

\* @desc This method will save or update a user in the USERS table

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem saving/updating the user

\*/

public bool SaveUser()

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// If this is a new user

if (this.Id\_user == -1)

{

// Create insert query

string query = "insert into users (id\_user, login, password, profile, active) " +

"values (NULL, '" + this.Login + "', MD5('" + this.Password +

"'), '" + this.Profile + "', '" + ((this.isActive) ? "1" : "0") + "')";

// Launch insert query

this.id\_user = conn.InsertToDB(query);

// Check result

if (this.id\_user > 0)

return true;

}

// This is an existing user

else

{

// Create update query

string query = "UPDATE users SET login = '" + this.Login + "', active = '" + ((this.isActive) ? "1" : "0") + "' "+

"WHERE id\_user = '" + this.Id\_user + "'";

// Launch update query

int result = conn.DeleteOrUpdate(query);

// Check update result

if (result > 0)

return true;

}

return false;

}

/\*\*

\* @desc Removes the user from the USERS table.

\* @params [none] No input parameter.

\* @return [bool] Returns true in case of success, false if there was problem deleting the user.

\*/

public bool DeleteUser()

{

// Create mysql connection

mySqlConn conn = new mySqlConn();

conn.connect();

// Create delete query

string query = "DELETE FROM users WHERE id\_user = '" + this.Id\_user + "'";

// Check delet query result

int result = conn.DeleteOrUpdate(query);

if (result > 0)

return true;

return false;

}

}

}

# Testing Methods

Software testing can obtain the quality of a product or service; it allows an independent view of the software and helps the developer understand the risks of implementing the software. The basic concept of software testing is the process of validating and verifying software of a program or application. Tests can check that the product meets the business and technical requirements of a company as shown by the original design. Software testing can take place at any time in the development process; however the most comprehensive testing takes place once all the coding has been completed. The gym system has been tested using Black Box, White Box and Usability Testing; the results are displayed in the following pages.

## Black Box Testing

Black Box Testing can be simply seen as the opposite of White Box Testing, the software is treated as a black box without any knowledge of the internal implementation of the software. This means that the user can use the software without knowing the limitations which can really push the performance of the software. For the first test a novice user will be asked to carry out two simple tasks that are both possible to see if they find problems that are not apparent to a user that has built the software.

Task 1: Add a member then book that member on a class.

Task 2: Add a class and a set of equipment to that class.

User Results

These are the observations of the user completing the tasks that were sent.

Task1: The user found it easy to login in and find the correct form very quickly, the new member was added within 2 minutes and the user didn’t have any queries on how to use the system. The user then started the second part of the task by looking for the class list, they opened a few forms before finding the correct form but once they did the double clicked the class they wished to enrol the member on and then checked that classes attendance list to check that it had worked. The user was able to do this without asking for any assistance; however it would be more efficient for the client if they were presented with a simple training manual so that if there were queries they could be simply resolved. The user is computer literate but has not used software like this before.

Task 2: This task involves the management functions of the software, the user logged in again and looked through the staff functions before then clicking on the management functions. Once the user opened the menu the user swiftly picked the correct ‘add class’ option, they then filled in the information and clicked the ‘equipment booking’ button to add a set of racket. When the user clicked save there was an error displayed as they had entered an incorrect ‘end time’. Once they closed the error and rectified the mistake and clicked save again. The user then decided to view the class list to view the information that they had added.

The user highlighted that the error message was precise allowing them to change the information that the page was rejecting. The user also had their own suggestion of adding messages incorporated to each window, for example if they could roll over an icon next to a button or field and information appear on the screen.

This kind of testing can be invaluable when trying to determine whether the software is ready to hand over to the client. An inexperienced user can give a very different perspective on the software and their suggestions can help improve the usability of the software.

## White Box Testing

This method of testing is completed by someone with the knowledge of the internal coding of the software. Test cases are then designed based on the internal structure as shown in the test cases on the following pages. The extensive testing of the software can determine whether or not the software meets the client’s requirements.

White Box -Test Cases

### Login

#### ‘Login’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Test username | ‘admin@ad.min’ | Combine with the correct password (“admin”) the system will log in. | Pass | 25/04/10 |
| 2 | Test username | ‘kmsmith’ | Log in will fail and an error message will be displayed. | Pass | 25/04/10 |
| 3 | Test password | ‘admin’ | Combined with the correct user name (“admin@ad.min”) the system will log in. | Pass | 25/04/10 |
| 4 | Test password | ‘jlk456’ | Log in will fail and an error message will be displayed. | Pass | 25/04/10 |
| 5 | Cancel button | Press ‘Cancel’ | Login window will close. | Pass | 25/04/10 |
| 6 | Accept button | Press ‘Accept’ | With correct data the login window will close and the menu will be displayed. | Pass | 25/04/10 |
| 7 | Accept button | Press ‘Accept’ | With incorrect data the login window will stay on display and an error message will be displayed. | Pass | 25/04/10 |

### Menu

#### ‘Menu’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Add New Member | Click Add New Member | A Blank Member form should be displayed to the user. |  | 24/04/10 |
| 2 | Edit Member | Click Edit Member | The member list should be displayed to the user so that they can search for a member they wish to edit. |  | 24/04/10 |
| 3 | Arrange Class | Click Arrange Class | The Class Arrange window will be displayed to that the used can fill in the blank booking form. |  | 24/04/10 |
| 4 | Edit Arranged Class | Click Edit Arranged Class | The Class Arrangements List will be displayed so that the users can the select a class they wish to edit. |  | 24/04/10 |
| 5 | Edit Equip Bookings | Click Edit Equip Bookings | The Equipment Booking List window is then displayed so that the users can pick a equipment booking that they wish to edit. |  | 24/04/10 |
| 6 | Edit Room List | Click Edit Room List | The room list is displayed so that the user can choose a room they wish to edit. |  | 24/04/10 |
| 7 | Management Functions | Click Management Functions | This expands the Management Functions Menu and shrinking the other menu that was displaying. |  | 24/04/10 |
| 8 | Add New Staff | Click Add New Staff | The Gym Staff Form window will open displaying a blank form that the user can fill out and save the information of a new member of staff. |  | 24/04/10 |
| 9 | Edit Staff | Click Edit Staff | The Staff List window will be displayed so that users can select members of staff that they wish to edit. |  | 24/04/10 |
| 10 | Membership Fees | Click Membership Fees | The Payments window will then open to show all the payments that have taken place, a new payment could then be added to the list. |  | 24/04/10 |
| 11 | Edit Arranged Class | Click Edit Arranged Class | The Class Arrangements List will then be displayed with the list of arranged classes, this classes can then be double clicked to open the class form where the information can be edited. |  | 24/04/10 |
| 12 | Add New Class | Click Add New Class | The Class Form window opens displaying a blank class information form. |  | 24/04/10 |
| 13 | Edit Class | Click Edit Class | The Class List window opens displaying the class list, the entries can then be double clicked and the information can be edited. |  | 24/04/10 |
| 14 | Add New Equipment | Click Add New Equipment | The Equipment Form window opens displaying a blank equipments form that can be filled in and saved to the equipment list. |  | 24/04/10 |
| 15 | Edit Equipment | Click Edit Equipment | The Equipment List window opens displaying the list of available equipments, each item can be doubled clicked the equipment form for that piece of equipment opens allowing users to edit this information and then save the changes. |  | 24/04/10 |
| 16 | Edit Room List | Click Edit Room List | The Room List window opens displaying the list of rooms than can then be double clicked to open the Room Form for that particular room containing the information about the room, the information can then be edited. The Room List also contains a ‘Add Room’ button. |  | 24/04/10 |
| 17 | Login Options | Click Login Options | This expands the Login Options Menu and shrinking the other menu that was displaying. |  | 24/04/10 |
| 18 | Change Password | Click Change Password | This opens the Password Update window that allows the user to change their password. |  | 24/04/10 |
| 19 | Staff Functions | Click Staff Functions | This expands the Staff Functions Menu and shrinking the other menu that was displaying. |  | 24/04/10 |

### Staff Functions

#### ‘Add New Member’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Member number | add several members | Check that a new member number is checked each time. | Pass | 25/04/10 |
| 2 | Member number | Try editing the number | The field will not be able to be edited. | Pass | 25/04/10 |
| 3 | Date of Birth | ‘19/06/1989’ | Date will be accepted and the user information added to the database. | Pass | 25/04/10 |
| 4 | Date of Birth | ‘14/04/87’ | Data will be rejected as it is the incorrect format, an error will be displayed. | FAIL | 25/04/10 |
| 5 | Date of Birth | ‘14/j4/5k6’ | Only integers should be able to be typed into this field. | Pass | 25/04/10 |
| 6 | Date of Birth | ‘12/56/1987’ | Data will be rejected as it is the incorrect format, an error will be displayed. | Pass | 25/04/10 |
| 7 | Postcode | ‘CB8 1HH’ | Data will be accepted and the user information added to the database. | Pass | 25/04/10 |
| 8 | Postcode | ‘GL20 8HL’ | Data will be accepted and the user information added to the database. | Pass | 25/04/10 |
| 9 | Postcode | ‘145 GHJ’ | Field is formatted to not let integers be typed in as the first character. | Pass | 25/04/10 |
| 10 | Telephone | ‘01223 456789’ | Data will be accepted and the user information added to the database. | Pass | 25/04/10 |
| 11 | Telephone | ‘01684 KL456’ | Data will not be save, an error will be displayed to the user when they click save. | FAIL | 25/04/10 |
| 12 | Email | ‘louiseplant@plant.com’ | Data will be accepted and the user information added to the database. | Pass | 25/04/10 |
| 13 | Email | ‘Louiseplant’ | Unless the information in the field contains ‘@’ and ‘.com’ it will produce an error when the user clicks save. | Pass | 25/04/10 |
| 14 | Medical Notes | Edit information | Data will be accepted and the user information added to the database. | Pass | 25/04/10 |
| 15 | Medical Notes  (Restore Button) | Click to restore data | All information in the box will be replaced with the original default. | Pass | 25/04/10 |
| 16 | Save & Open | Correct data | If all data is correct then window will appear notifying the user that the member has been added successfully. The member list will then be displayed. | Pass | 25/04/10 |
| 17 | Save & Open | Incorrect data | A window will appear highlighting errors within the form that need to be corrected. The Member information member will stay displayed so that the errors can be resolved. | Pass | 25/04/10 |
| 18 | Save & Close | Correct data | If all data is correct then window will appear notifying the user that the member has been added successfully. The page will then close. | Pass | 25/04/10 |
| 19 | Save & Close | Incorrect data | A window will appear highlighting errors within the form that need to be corrected. The Member information member will stay displayed so that the errors can be resolved. | Pass | 25/04/10 |
| 20 | Save & Stay | Correct data | If all data is correct then window will appear notifying the user that the member information has been saved successfully. It will then stay on that Members Information page. | Pass | 25/04/10 |
| 21 | Save & Stay | Incorrect data | A window will appear highlighting errors within the form that need to be corrected. The Member information member will stay displayed so that the errors can be resolved. | Pass | 25/04/10 |

#### ‘Edit Member’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Membership Type | ‘Student | Display all the members that have the Membership Type ‘Student’. | FAIL | 25/04/10 |
| 2 | First Name | ‘Louise’ | Displays the members with the first name ‘Louise’ i.e Member Number 37. | Pass | 25/04/10 |
| 3 | First Name | ‘John’ | No members are displayed as there are no users with the first name ‘John’. | Pass | 25/04/10 |
| 4 | Last Name | ‘Plant’ | Displays all the members the last name ‘Plant’ i.e Member Number 37. | Pass | 25/04/10 |
| 5 | Last Name | ‘Smith’ | No members are displayed as there are no users with the last name ‘Smith’. | Pass | 25/04/10 |
| 6 | Date of Birth | ‘12/07/1990’ | Displays all the members the date of birth ‘12/07/1990’. i.e Member Number 17. | Pass | 25/04/10 |
| 7 | Date of Birth | ‘01/05/1989’ | No members are displayed as there are no users with the date of birth ‘01/05/1989’. | Pass | 25/04/10 |
| 8 | Email | ‘louiseplant@plant.com’ | Displays the members with the email ‘louiseplant@plant.com’ i.e Member Number 37. | Pass | 25/04/10 |
| 9 | Email | ‘john@yahoo.co.uk’ | No members are displayed as there are no users with the email ‘john@yahoo.co.uk’. | Pass | 25/04/10 |
| 10 | Search | Click Search | When the Search is clicked it will run the criteria though the database and return the matching results. | Pass | 25/04/10 |
| 11 | Add New Member | Click Add New Member | The Member Form window will open displaying a blank Member Form. | Pass | 25/04/10 |
| 12 | Member Results | Double Click Result | Member Information Form will open displaying the current member information that can then be edited and the changes saved. | Pass | 25/04/10 |
| 13 | Copy All Selected Cells To Clipboard | Click Copy All Selected Cells To Clipboard | The selected will be saved to the clipboard and can then be pasted in other programs. i.e Microsoft Excel. | Pass | 25/04/10 |

#### ‘Arrange Class’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Date | ‘14/04/2010’ | Data will be accepted and the class information added to the database. | Pass | 25/04/10 |
| 2 | Date | Blank | The validation will produce an error instructor the user of the error within the form. | Pass | 25/04/10 |
| 3 | Start Time | ‘03.00’ | Data will be accepted and the class information added to the database. | Pass | 25/04/10 |
| 4 | Start Time | Blank | The validation will produce an error instructor the user of the error within the form. | Pass | 25/04/10 |
| 5 | End Time | ‘04.00’ | Data will be accepted and the class information added to the database. | Pass | 25/04/10 |
| 6 | End Time | Blank | The validation will produce an error instructor the user of the error within the form. | Pass | 25/04/10 |
| 7 | Save & Open | Click Save & Open | If the information is correct then it will be save to the Class List, the Class List will then be displayed on the screen. | Pass | 25/04/10 |
| 8 | Save & Open | Click Save & Open | If the information is incorrect then the validation will produce errors and display a window to instruct the user what to do. | Pass | 25/04/10 |
| 9 | Save & Stay | Click Save & Stay | If the information is correct then it will be save to the Class List, and the Class Information form will stay displayed on the screen. | Pass | 25/04/10 |
| 10 | Save & Stay | Click Save & Stay | If the information is incorrect then the validation will produce errors and display a window to instruct the user what to do. | Pass | 25/04/10 |
| 11 | Save & Close | Click Save & Close | If the information is correct then it will be save to the Class List, the Class List form will then be closed. | Pass | 25/04/10 |
| 12 | Save & Close | Click Save & Close | If the information is incorrect then the validation will produce errors and display a window to instruct the user what to do. | Pass | 25/04/10 |
| 13 | Equipment Booking | Click Equipment Booking | The Equipment List window will open showing the current equipment bookings. | Pass | 25/04/10 |
| 14 | View Attendants | Click View Attendants | The Members List Form will open showing the current list of Members attending that class. | Pass | 25/04/10 |
| 15 | Enroll Members | Click Enroll Members | The Members List window will then open, Members can then be chosen to add to the class list. | Pass | 25/04/10 |

#### ‘Edit Arranged Class’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Search | ‘Yoga’ | A list of all the Yoga classes will be returned to the window. Any of the results can then be double clicked to edit the class information. | Pass | 25/04/10 |
| 2 | Search | ‘Track’ | No results will be returned as there is no class by the name of ‘Track’. | Pass | 25/04/10 |

#### ‘Edit Equip Bookings’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Edit Booking | Double Click a Booking | A window will be displayed so that equipment can be returned. | Pass | 25/04/10 |
| 2 | Return Equipment | Return a football (Booking Number 16) | Once the booking has been selected and the return window has opened then the user can select how many items they wish to return and the equipment list will be updated. | Pass | 25/04/10 |

#### ‘Edit Room List’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Search | ‘3’ | The information for room number ‘3’ is returned to the window. | Pass | 25/04/10 |
| 2 | Add Room | Click Add Room | A Blank Room form will be displayed so that a user can fill it in and add the new room will be added to the Room List. | Pass | 25/04/10 |
| 3 | Edit Room Information | Double Click Room 3 | The Room Form containing that room’s information will be displayed; it can then be edited and saved. The list will then also be updated. | Pass | 25/04/10 |
| 4 | Remove Room | Click Remove | In the Room Form that opens when double clicking a room on the room list there is a button at the bottom of the window that will remove the room from the list. | Pass | 25/04/10 |

### Managements

#### ‘Add New Staff’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Date of Birth | ‘24/06/1987’ | Date will be accepted and the user information added to the database. | Pass | 26/04/10 |
| 2 | Date of Birth | ‘31/04/87’ | Data will be rejected as it is the incorrect format, an error will be displayed. | FAIL | 26/04/10 |
| 3 | Date of Birth | ‘18j4/546’ | Only integers should be able to be typed into this field. | Pass | 26/04/10 |
| 4 | Date of Birth | ‘12/99/1786’ | Data will be rejected as it is the incorrect format, an error will be displayed. | Pass | 26/04/10 |
| 5 | Postcode | ‘CB8 1HH’ | Data will be accepted and the user information added to the database. | Pass | 26/04/10 |
| 6 | Postcode | ‘GL20 8HL’ | Data will be accepted and the user information added to the database. | Pass | 26/04/10 |
| 7 | Postcode | ‘145 GHJ’ | Field is formatted to not let integers be typed in as the first character. | Pass | 26/04/10 |
| 8 | Telephone | ‘01223 456789’ | Data will be accepted and the user information added to the database. | Pass | 26/04/10 |
| 9 | Telephone | ‘01684 KL456’ | Data will not be save, an error will be displayed to the user when they click save. | FAIL | 26/04/10 |
| 10 | Email | ‘tomshoe@plant.com’ | Data will be accepted and the user information added to the database. | Pass | 26/04/10 |
| 11 | Email | ‘tomshoe’ | Unless the information in the field contains ‘@’ and ‘.com’ it will produce an error when the user clicks save. | Pass | 26/04/10 |
| 12 | Contract Start | Automatic data | When the Gym Staff Form is opened the current date is automatically generated and placed in the Contract Start field. | Pass | 26/04/10 |
| 13 | Contract Start | ‘10/jk/4’ | Field validation will not allow for anything integers to typed into the field | Pass | 26/04/10 |
| 14 | Contract Finish | ‘20/05/2011’ | The data should be accepted and saved to the Staff list within the database. | Pass | 26/04/10 |
| 15 | Contract Finish | ‘14/06/10kl’ | Field validation will not allow for anything integers to typed into the field | Pass | 26/04/10 |
| 16 | Employment Type | Drop Down List | The list contains the correct options and allows the user to select the appropriate choice. | Pass | 26/04/10 |
| 17 | Position | Drop Down List | The list contains the correct options and allows the user to select the appropriate choice. | Pass | 26/04/10 |
| 18 | Remove | Click Remove | This will delete the current Member of staff. | Pass | 26/04/10 |
| 19 | Cancel | Click Cancel | Will close the window saving no changes to the databse. | Pass | 26/04/10 |
| 20 | Save & Open | Correct data | If all data is correct then window will appear notifying the user that the member has been added successfully. The member list will then be displayed. | Pass | 25/04/10 |
| 21 | Save & Open | Incorrect data | A window will appear highlighting errors within the form that need to be corrected. The Member information member will stay displayed so that the errors can be resolved. | Pass | 25/04/10 |
| 22 | Save & Close | Correct data | If all data is correct then window will appear notifying the user that the member has been added successfully. The page will then close. | Pass | 25/04/10 |
| 23 | Save & Close | Incorrect data | A window will appear highlighting errors within the form that need to be corrected. The Member information member will stay displayed so that the errors can be resolved. | Pass | 25/04/10 |
| 24 | Save & Stay | Correct data | If all data is correct then window will appear notifying the user that the member information has been saved successfully. It will then stay on that Members Information page. | Pass | 25/04/10 |
| 25 | Save & Stay | Incorrect data | A window will appear highlighting errors within the form that need to be corrected. The Member information member will stay displayed so that the errors can be resolved. | Pass | 25/04/10 |

#### ‘Edit Staff’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | First Name | ‘George’ | Displays the members with the first name ‘George’ i.e. Staff Number 6. | FAIL | 26/04/10 |
| 2 | First Name | ‘Peter’ | No members are displayed as there are no users with the first name ‘Peter’. | FAIL | 26/04/10 |
| 3 | Last Name | ‘Clooney’ | Displays the members with the first name ‘Clooney’ i.e. Staff Number 6. | FAIL | 26/04/10 |
| 4 | Last Name | ‘Beaver’ | No members are displayed as there are no users with the last name ‘Beaver’. | FAIL | 26/04/10 |
| 5 | Date of Birth | ‘01/01/1999’ | Displays all the members the date of birth ‘01/01/1999’. i.e Member Number 6. | FAIL | 26/04/10 |
| 6 | Date of Birth | ‘01/05/1989’ | No members are displayed as there are no users with the date of birth ‘01/05/1989’. | FAIL | 26/04/10 |
| 7 | Email | ‘janimani@a.com’ | Displays the members with the email ‘janimani@a.com’ i.e Member Number 1. | FAIL | 26/04/10 |
| 8 | Email | ‘jane@yahoo.co.uk’ | No members are displayed as there are no users with the email ‘jane@yahoo.co.uk’. | FAIL | 26/04/10 |
| 9 | Search | Click Search | When the Search is clicked it will run the criteria though the database and return the matching results. | FAIL | 26/04/10 |
| 10 | Add Staff Member | Click Add New Member | The Staff Form window will open displaying a blank Staff Form that can be filled in a saved. | Pass | 26/04/10 |
| 11 | Close | Click Close | The current window will close. | Pass | 26/04/10 |

#### ‘Membership Fees’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Add Payments | Click Add Payment | The Member List will open. | Pass | 26/04/10 |
| 2 | Add Payment | Choose Member | Double clicking on a member will open the Add Payment Form, this form can then be filled out to record the payment details which will then be added to the payment list. | Pass | 26/04/10 |

#### ‘Edit Arranged Class’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Search | ‘Yoga’ | Three results will be displayed showing the Yoga classes. | Pass | 26/04/10 |
| 2 | Edit Class | Double Click a Class | The Class Arrange window will open, all the information can then be edited and saved. | Pass | 26/04/10 |
| 3 | Equipment Bookings | Click Equipment Bookings | This will open the Equipment List allowing users to add equipment to the class. | Pass | 26/04/10 |
| 4 | Add Equipment | Click on a item of Equipment | A window will open asking if you wish to borrow the piece of equipment or borrow it. | Pass | 26/04/10 |
| 5 | View Attendants | Click View Attendants | This will open the Members list showing all the members attending that class. | Pass | 26/04/10 |
| 6 | Enroll Members | Click Enroll Members | This will open the Member List Form that contains the list of all Member’s, the members can then be searched for and added to the class. | Pass | 26/04/10 |
| 7 | Enroll Member | Enroll Member on a Class | Once the user double clicks on the member a window will pop up asking whether the user wants to add this Member to the class. | Pass | 26/04/10 |

#### ‘Add New Class’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Add Class | ‘Fitness’, ‘1hour.’ Group | The class will be added to the class list. | Pass | 26/04/10 |
| 2 | Save | Click Save | This information will be saved and the class will appear on the Class List. | Pass | 26/04/10 |

#### ‘Edit Class’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Search | ‘Fitness’ | The Fitness classes are then displayed in the window. | Pass | 26/04/10 |
| 2 | View Class Information | Double click Fitness | The Class Information window will then appear displaying that class information. | Pass | 26/04/10 |
| 3 | Add Class | ‘Dance’ | A blank Class Form will open so that the user can fill it in. | Pass | 26/04/10 |
| 4 | Save | Click Save | The information will then be saved and the class added to the Class List | Pass | 26/04/10 |
| 5 | Cancel | Click Cancel | The window will close. | Pass | 26/04/10 |

#### ‘Add New Equipment’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Radio Buttons | ‘Set’ | The Set Name box and other drop down lists will activate to be able to log the information for the set. | Pass | 26/04/10 |
| 2 | Radio Buttons | ‘Item’ | The Item Name box and Description will then become active allowing the user to save the correct information. | Pass | 26/04/10 |
| 3 | Save | ‘Racket’ | Information will be saved to the database and appear on the Equipment List. | Pass | 26/04/10 |
| 4 | Close | Click Close | Will close the current window. | Pass | 26/04/10 |

#### ‘Edit Equipment’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Radio Buttons | ‘Set’ | The list of Sets will show. | Pass | 26/04/10 |
| 2 | Radio Buttons | ‘Item’ | The list of Items will show. | Pass | 26/04/10 |
| 3 | Add Equipment | Click Add Equipment | The Equipment Form will open allowing the user to add a new item or a new set. | Pass | 26/04/10 |
| 4 | Edit Equipment | Double Click Set ‘Rackets’ | The Equipment Form containing that set information. | Pass | 26/04/10 |
| 5 | Remove | Click Remove | This will delete the current Item/Set. | Pass | 26/04/10 |
| 6 | Save | Change description ‘red’ | Information will be saved to the database and appear on the Equipment List. | Pass | 26/04/10 |
| 7 | Close | Click Close | Will close the current window. | Pass | 26/04/10 |

#### ‘Edit Room List’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Search | ‘RT123’ | The results will display to screen. | Pass | 26/04/10 |
| 2 | Search | ‘Bry013’ | No results will be returned as there is no room with the name Bry013. | Pass | 26/04/10 |
| 4 | Add Room | Click Room Form | A blank room form will open allowing the user to enter the information for a new room. | Pass | 26/04/10 |
| 5 | Save | ‘TennisCourt’ | The room will be added to the Room List. | Pass | 26/04/10 |
| 6 | Remove | Remove RT123 | Double click the room entry then click remove and a window will the pop up asking you to confirm the operation before removing the room from the list and the database. | Pass | 26/04/10 |
| 7 | Cancel | Click Cancel | The current window will close. | Pass | 26/04/10 |

### Login Option

#### ‘Change Password’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Old Password | blank | An error message will appear instructor the user to enter the correct information. | Pass | 26/04/10 |
| 2 | New Password | blank | An error message will appear instructor the user to enter the correct information. | Pass | 26/04/10 |
| 3 | Repeat Password | blank | An error message will appear instructor the user to enter the correct information. | Pass | 26/04/10 |
| 4 | New Password | ‘beetle’ | If the passwords don’t match an error will display on screen. | Pass | 26/04/10 |
| Repeat Password | ‘red’ |

## Usability Testing

Usability Test Cases

### 1. Visibility of System Status

The system should always keep user informed about what is going on, through appropriate feedback within reasonable time.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 1.1 | Does every display begin with a title or header that describes screen contents? | X O O | 01/May/2010 |
| 1.2 | Is there a consistent icon design scheme and stylistic treatment across the system? | X O O | 01/May/2010 |
| 1.3 | Is a single, selected icon clearly visible when surrounded by unselected icons? | X O O | 01/May/2010 |
| 1.4 | Do menu instructions, prompts, and error messages appear in the same place(s) on each menu? | X O O | 01/May/2010 |
| 1.5 | In multipage data entry screens, is each page labeled to show its relation to others? | X O O | 01/May/2010 |
| 1.6 | If overtype and insert mode are both available, is there a visible indication of which one the user is in? | O O X | 01/May/2010 |
| 1.7 | If pop-up windows are used to display error messages, do they allow the user to see the field in error? | X O O | 01/May/2010 |
| 1.8 | Is there some form of system feedback for every operator action? | X O O | 01/May/2010 |
| 1.9 | After the user completes an action (or group of actions), does the feedback indicate that the next group of actions can be started? | X O O | 01/May/2010 |
| 1.10 | Is there visual feedback in menus or dialog boxes about which choices are selectable? | X O O | 01/May/2010 |
| 1.11 | Is there visual feedback in menus or dialog boxes about which choice the cursor is on now? | X O O | 01/May/2010 |
| 1.12 | If multiple options can be selected in a menu or dialog box, is there visual feedback about which options are already selected? | X O O | 01/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 1.13 | Is there visual feedback when objects are selected or moved? | X O O | 01/May/2010 |
| 1.14 | Is the current status of an icon clearly indicated? | X O O | 01/May/2010 |
| 1.15 | Is there feedback when function keys are pressed? | O O X | 01/May/2010 |
| 1.16 | If there are observable delays (greater than fifteen seconds) in the system’s response time, is the user kept informed of the system's progress? | O O X | 01/May/2010 |
| 1.17 | Are response times appropriate to the task? | X O O | 01/May/2010 |
| 1.18 | Typing, cursor motion, mouse selection: 50-1 50 milliseconds | X O O | 01/May/2010 |
| 1.19 | Simple, frequent tasks: less than 1 second | X O O | 01/May/2010 |
| 1.20 | Common tasks: 2-4 seconds | X O O | 01/May/2010 |
| 1.21 | Complex tasks: 8-12 seconds | X O O | 01/May/2010 |
| 1.22 | Are response times appropriate to the user's cognitive processing? | X O O | 01/May/2010 |
| 1.23 | Continuity of thinking is required and information must be remembered throughout several responses: less than two seconds. | X O O | 01/May/2010 |
| 1.24 | High levels of concentration aren't necessary and remembering information is not required: two to fifteen seconds. | X O O | 01/May/2010 |
| 1.25 | Is the menu-naming terminology consistent with the user's task domain? | X O O | 01/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 1.26 | Does the system provide *visibility:* that is, by looking, can the user tell the state of the system and the alternatives for action? | X O O | 01/May/2010 |
| 1.27 | Do GUI menus make obvious which item has been selected? | X O O | 01/May/2010 |
| 1.28 | Do GUI menus make obvious whether deselection is possible? | O O X | 01/May/2010 |
| 1.29 | If users must navigate between multiple screens, does the system use context labels, menu maps, and place markers as navigational aids? | O O X | 01/May/2010 |

### 2. Match Between System and the Real World

The system should speak the user’s language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 2.1 | Are icons concrete and familiar? | X O O | 01/May/2010 |
| 2.2 | Are menu choices ordered in the most logical way, given the user, the item names, and the task variables? | X O O | 01/May/2010 |
| 2.3 | If there is a natural sequence to menu choices, has it been used? | X O O | 01/May/2010 |
| 2.4 | Do related and interdependent fields appear on the same screen? | X O O | 01/May/2010 |
| 2.5 | If shape is used as a visual cue, does it match cultural conventions? | X O O | 01/May/2010 |
| 2.6 | Do the selected colors correspond to common expectations about color codes? | X O O | 01/May/2010 |
| 2.7 | When prompts imply a necessary action, are the words in the message consistent with that action? | X O O | 01/May/2010 |
| 2.8 | Do keystroke references in prompts match actual key names? | X O O | 01/May/2010 |
| 2.9 | On data entry screens, are tasks described in terminology familiar to users? | X O O | 01/May/2010 |
| 2.10 | Are field-level prompts provided for data entry screens? | X O O | 01/May/2010 |
| 2.11 | For question and answer interfaces, are questions stated in clear, simple language? | X O O | 01/May/2010 |
| 2.12 | Do menu choices fit logically into categories that have readily understood meanings? | X O O | 01/May/2010 |
| 2.13 | Are menu titles parallel grammatically? | X O O | 01/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 2.14 | Does the command language employ user jargon and avoid computer jargon? | X O O | 01/May/2010 |
| 2.15 | Are command names specific rather than general? | X O O | 01/May/2010 |
| 2.16 | Does the command language allow both full names and abbreviations? | X O O | 01/May/2010 |
| 2.17 | Are input data codes meaningful? | X O O | 01/May/2010 |
| 2.18 | Have uncommon letter sequences been avoided whenever possible? | X O O | 01/May/2010 |
| 2.19 | Does the system automatically enter leading or trailing spaces to align decimal points? | O X O | 01/May/2010 |
| 2.20 | Does the system automatically enter a dollar sign and decimal for monetary entries? | O O X | 01/May/2010 |
| 2.21 | Does the system automatically enter commas in numeric values greater than 9999? | O X O | 01/May/2010 |
| 2.22 | Do GUI menus offer activation: that is, make obvious how to say *“now do it"?* | X O O | 01/May/2010 |
| 2.23 | Has the system been designed so that keys with similar names do not perform opposite (and potentially dangerous) actions? | O O X | 01/May/2010 |
| 2.24 | Are function keys labeled clearly and distinctively, even if this means breaking consistency rules? | X O O | 01/May/2010 |

### 3. User Control and Freedom

Users should be free to select and sequence tasks (when appropriate), rather than having the system do this for them. Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Users should make their own decisions (with clear information) regarding the costs of exiting current work. The system should support undo and redo.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 3.1 | If setting up windows is a low-frequency task, is it particularly easy to remember? | X O O | 01/May/2010 |
| 3.2 | In systems that use overlapping windows, is it easy for users to rearrange windows on the screen? | O O X | 01/May/2010 |
| 3.3 | In systems that use overlapping windows, is it easy for users to switch between windows? | O O X | 01/May/2010 |
| 3.4 | When a user's task is complete, does the system wait for a signal from the user before processing? | X O O | 01/May/2010 |
| 3.5 | Can users type-ahead in a system with many nested menus? | O X O | 01/May/2010 |
| 3.6 | Are users prompted to confirm commands that have drastic, destructive consequences? | O X O | 01/May/2010 |
| 3.7 | Is there an "undo" function at the level of a single action, a data entry, and a complete group of actions? | O X O | 01/May/2010 |
| 3.8 | Can users cancel out of operations in progress? | X O O | 01/May/2010 |
| 3.9 | Are character edits allowed in commands? | O O X | 01/May/2010 |
| 3.10 | Can users reduce data entry time by copying and modifying existing data? | X O O | 01/May/2010 |
| 3.11 | Are character edits allowed in data entry fields? | X O O | 01/May/2010 |
| 3.12 | If menu lists are long (more than seven items), can users select an item either by moving the cursor or by typing a mnemonic code? | O O X | 01/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 3.13 | If the system uses a pointing device, do users have the option of either clicking on menu items or using a keyboard shortcut? | O X O | 01/May/2010 |
| 3.14 | Are menus broad (many items on a menu) rather than deep (many menu levels)? | X O O | 01/May/2010 |
| 3.15 | If the system has multiple menu levels, is there a mechanism that allows users to go back to previous menus? | X O O | 01/May/2010 |
| 3.16 | If users can go back to a previous menu, can they change their earlier menu choice? | X O O | 01/May/2010 |
| 3.17 | Can users move forward and backward between fields or dialog box options? | X O O | 01/May/2010 |
| 3.18 | If the system has multipage data entry screens, can users move backward and forward among all the pages in the set? | O O X | 01/May/2010 |
| 3.19 | If the system uses a question and answer interface, can users go back to previous questions or skip forward to later questions? | O O X | 01/May/2010 |
| 3.20 | Do function keys that can cause serious consequences have an undo feature? | O O X | 01/May/2010 |
| 3.21 | Can users easily reverse their actions? | O X O | 01/May/2010 |
| 3.22 | If the system allows users to reverse their actions, is there a retracing mechanism to allow for multiple undos? | O O X | 01/May/2010 |
| 3.23 | Can users set their own system, session, file, and screen defaults? | O X O | 01/May/2010 |

### 4. Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 4.1 | Have industry or company formatting standards been followed consistently in all screens within a system? | X O O | 01/May/2010 |
| 4.2 | Has a heavy use of all uppercase letters on a screen been avoided? | X O O | 01/May/2010 |
| 4.3 | Do abbreviations not include punctuation? | X O O | 01/May/2010 |
| 4.4 | Are integers right-justified and real numbers decimal-aligned? | X O O | 01/May/2010 |
| 4.5 | Are icons labeled? | X O O | 01/May/2010 |
| 4.6 | Are there no more than twelve to twenty icon types? | X O O | 01/May/2010 |
| 4.7 | Are there salient visual cues to identify the active window? | X O O | 01/May/2010 |
| 4.8 | Does each window have a title? | X O O | 01/May/2010 |
| 4.9 | Are vertical and horizontal scrolling possible in each window? | X O O | 01/May/2010 |
| 4.10 | Does the menu structure match the task structure? | X O O | 01/May/2010 |
| 4.11 | Have industry or company standards been established for menu design, and are they applied consistently on all menu screens in the system? | X O O | 01/May/2010 |
| 4.12 | Are menu choice lists presented vertically? | O O O | 01/May/2010 |
| 4.13 | If "exit" is a menu choice, does it always appear at the bottom of the list? | O O X | 01/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 4.14 | Are menu titles either centered or left-justified? | O O X | 01/May/2010 |
| 4.15 | Are menu items left-justified, with the item number or mnemonic preceding the name? | O O X | 01/May/2010 |
| 4.16 | Do embedded field-level prompts appear to the right of the field label? | O O X | 01/May/2010 |
| 4.17 | Do on-line instructions appear in a consistent location across screens? | O O X | 01/May/2010 |
| 4.18 | Are field labels and fields distinguished typographically? | X O O | 01/May/2010 |
| 4.19 | Are field labels consistent from one data entry screen to another? | X O O | 01/May/2010 |
| 4.20 | Are fields and labels left-justified for alpha lists and right-justified for numeric lists? | X O O | 01/May/2010 |
| 4.21 | Do field labels appear to the left of single fields and above list fields? | X O O | 01/May/2010 |
| 4.22 | Are attention-getting techniques used with care? | X O O | 01/May/2010 |
| 4.23 | Intensity: two levels only | X O O | 01/May/2010 |
| 4.24 | Size: up to four sizes | X O O | 01/May/2010 |
| 4.25 | Font: up to three | X O O | 01/May/2010 |
| 4.26 | Blink: two to four hertz | O O X | 01/May/2010 |
| 4.27 | Color: up to four (additional colors for occasional use only) | X O O | 01/May/2010 |
| 4.28 | Sound: soft tones for regular positive feedback, harsh for rare critical conditions | O O X | 01/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 4.29 | Are attention-getting techniques used only for exceptional conditions or for time-dependent information? | X O O | 01/May/2010 |
| 4.30 | Are there no more than four to seven colors, and are they far apart along the visible spectrum? | X O O | 01/May/2010 |
| 4.31 | Is a legend provided if color codes are numerous or not obvious in meaning? | O O X | 01/May/2010 |
| 4.32 | Have pairings of high-chroma, spectrally extreme colors been avoided? | X O O | 01/May/2010 |
| 4.33 | Are saturated blues avoided for text or other small, thin line symbols? | X O O | 01/May/2010 |
| 4.34 | Is the most important information placed at the beginning of the prompt? | X O O | 01/May/2010 |
| 4.35 | Are user actions named consistently across all prompts in the system? | X O O | 01/May/2010 |
| 4.36 | Are system objects named consistently across all prompts in the system? | X O O | 01/May/2010 |
| 4.37 | Do field-level prompts provide more information than a restatement of the field name? | X O O | 01/May/2010 |
| 4.38 | For question and answer interfaces, are the valid inputs for a question listed? | X O O | 01/May/2010 |
| 4.39 | Are menu choice names consistent, both within each menu and across the system, in grammatical style and terminology? | X O O | 01/May/2010 |
| 4.40 | Does the structure of menu choice names match their corresponding menu titles? | X O O | 01/May/2010 |
| 4.41 | Are commands used the same way, and do they mean the same thing, in all parts of the system? | X O O | 01/May/2010 |
| 4.42 | Does the command language have a consistent, natural, and mnemonic syntax? | X O O | 01/May/2010 |

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 4.43 | Do abbreviations follow a simple primary rule and, if necessary, a simple secondary rule for abbreviations that otherwise would be duplicates? | X O O | 01/May/2010 |
| 4.44 | Is the secondary rule used only when necessary? | O O O | 01/May/2010 |
| 4.45 | Are abbreviated words all the same length? | O O O | 01/May/2010 |
| 4.46 | Is the structure of a data entry value consistent from screen to screen? | O O O | 01/May/2010 |
| 4.47 | Is the method for moving the cursor to the next or previous field consistent throughout the system? | O O O | 01/May/2010 |
| 4.48 | If the system has multipage data entry screens, do all pages have the same title? | O O O | 01/May/2010 |
| 4.49 | If the system has multipage data entry screens, does each page have a sequential page number? | O O O | 01/May/2010 |
| 4.50 | Does the system follow industry or company standards for function key assignments? | O O O | 01/May/2010 |
| 4.51 | Are high-value, high-chroma colors used to attract attention? | O O O | 01/May/2010 |

### 5. Help Users Recognize, Diagnose, and Recover From Errors

Error messages should be expressed in plain language(NO CODES).

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 5.1 | Is sound used to signal an error? | X O O | 02/May/2010 |
| 5.2 | Are prompts stated constructively, without overt or implied criticism of the user? | X O O | 02/May/2010 |
| 5.3 | Do prompts imply that the user is in control? | X O O | 02/May/2010 |
| 5.4 | Are prompts brief and unambiguous. | X O O | 02/May/2010 |
| 5.5 | Are error messages worded so that the system, not the user, takes the blame? | X O O | 02/May/2010 |
| 5.6 | If humorous error messages are used, are they appropriate and inoffensive to the user population? | O O X | 02/May/2010 |
| 5.7 | Are error messages grammatically correct? | X O O | 02/May/2010 |
| 5.8 | Do error messages avoid the use of exclamation points? | O X O | 02/May/2010 |
| 5.9 | Do error messages avoid the use of violent or hostile words? | X O O | 02/May/2010 |
| 5.10 | Do error messages avoid an anthropomorphic tone? | X O O | 02/May/2010 |
| 5.11 | Do all error messages in the system use consistent grammatical style, form, terminology, and abbreviations? | X O O | 02/May/2010 |
| 5.12 | Do messages place users in control of the system? | X O O | 02/May/2010 |
| 5.13 | Does the command language use normal action-object syntax? | O O X | 02/May/2010 |

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 5.14 | Does the command language avoid arbitrary, non-English use of punctuation, except for symbols that users already know? | O O X | 02/May/2010 |
| 5.15 | If an error is detected in a data entry field, does the system place the cursor in that field or highlight the error? | O X O | 02/May/2010 |
| 5.16 | Do error messages inform the user of the error's severity? | X O O | 02/May/2010 |
| 5.17 | Do error messages suggest the cause of the problem? | X O O | 02/May/2010 |
| 5.18 | Do error messages provide appropriate semantic information? | X O O | 02/May/2010 |
| 5.19 | Do error messages provide appropriate syntactic information? | X O O | 02/May/2010 |
| 5.20 | Do error messages indicate what action the user needs to take to correct the error? | X O O | 02/May/2010 |
| 5.21 | If the system supports both novice and expert users, are multiple levels of error-message detail available? | O X O | 02/May/2010 |

### 6. Error Prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 6.1 | If the database includes groups of data, can users enter more than one group on a single screen? | X O O | 02/May/2010 |
| 6.2 | Have dots or underscores been used to indicate field length? | O X O | 02/May/2010 |
| 6.3 | Is the menu choice name on a higher-level menu used as the menu title of the lower-level menu? | X O O | 02/May/2010 |
| 6.4 | Are menu choices logical, distinctive, and mutually exclusive? | X O O | 02/May/2010 |
| 6.5 | Are data inputs case-blind whenever possible? | X O O | 02/May/2010 |
| 6.6 | If the system displays multiple windows, is navigation between windows simple and visible? | O O X | 02/May/2010 |
| 6.7 | Are the function keys that can cause the most serious consequences in hard-to-reach positions? | O O X | 02/May/2010 |
| 6.8 | Are the function keys that can cause the most serious consequences located far away from low-consequence and high-use keys? | O O X | 02/May/2010 |
| 6.9 | Has the use of qualifier keys been minimized? | O O X | 02/May/2010 |
| 6.10 | If the system uses qualifier keys, are they used consistently throughout the system? | O O X | 02/May/2010 |
| 6.11 | Does the system prevent users from making errors whenever possible? | X O O | 02/May/2010 |
| 6.12 | Does the system warn users if they are about to make a potentially serious error? | O X O | 02/May/2010 |
| 6.13 | Does the system intelligently interpret variations in user commands? | O O X | 02/May/2010 |
| 6.14 | Do data entry screens and dialog boxes indicate the number of character spaces available in a field? | O X O | 02/May/2010 |
| 6.15 | Do fields in data entry screens and dialog boxes contain default values when appropriate? | X O O | 02/May/2010 |

### 7. Recognition Rather Than Recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 7.1 | For question and answer interfaces, are visual cues and white space used to distinguish questions, prompts, instructions, and user input? | X O O | 02/May/2010 |
| 7.2 | Does the data display start in the upper-left corner of the screen? | X O O | 02/May/2010 |
| 7.3 | Are multiword field labels placed horizontally (not stacked vertically)? | X O O | 02/May/2010 |
| 7.4 | Are all data a user needs on display at each step in a transaction sequence? | X O O | 02/May/2010 |
| 7.5 | Are prompts, cues, and messages placed where the eye is likely to be looking on the screen? | X O O | 02/May/2010 |
| 7.6 | Have prompts been formatted using white space, justification, and visual cues for easy scanning? | X O O | 02/May/2010 |
| 7.7 | Do text areas have "breathing space" around them? | X O O | 02/May/2010 |
| 7.8 | Is there an obvious visual distinction made between "choose one" menu and "choose many" menus? | X O O | 02/May/2010 |
| 7.9 | Have spatial relationships between soft function keys (on-screen cues) and keyboard function keys been preserved? | O O X | 02/May/2010 |
| 7.10 | Does the system gray out or delete labels of currently inactive soft function keys? | X O O | 02/May/2010 |
| 7.11 | Is white space used to create symmetry and lead the eye in the appropriate direction? | X O O | 02/May/2010 |
| 7.12 | Have items been grouped into logical zones, and have headings been used to distinguish between zones? | X O O | 02/May/2010  No headings though. |
| # | Review Checklist | Yes No N/A | Checked on |
| 7.13 | Are zones no more than twelve to fourteen characters wide and six to seven lines high? | X O O | 02/May/2010 |
| 7.14 | Have zones been separated by spaces, lines, color, letters, bold titles, rules lines, or shaded areas? | X O O | 02/May/2010 |
| 7.15 | Are field labels close to fields, but separated by at least one space? | X O O | 02/May/2010 |
| 7.16 | Are long columnar fields broken up into groups of five, separated by a blank line? | O O X | 02/May/2010 |
| 7.17 | Are optional data entry fields clearly marked? | X O O | 02/May/2010 |
| 7.18 | Are symbols used to break long input strings into "chunks"? | O O X | 02/May/2010 |
| 7.19 | Is reverse video or color highlighting used to get the user's attention? | O X O | 02/May/2010 |
| 7.20 | Is reverse video used to indicate that an item has been selected? | X O O | 02/May/2010 |
| 7.21 | Are size, boldface, underlining, color, shading, or typography used to show relative quantity or importance of different screen items? | X O O | 02/May/2010 |
| 7.22 | Are borders used to identify meaningful groups? | O X O | 02/May/2010 |
| 7.23 | Has the same color been used to group related elements? | O O X | 02/May/2010 |
| 7.24 | Is color coding consistent throughout the system? | X O O | 02/May/2010 |
| 7.25 | Is color used in conjunction with some other redundant cue? | O O X | 02/May/2010 |
| 7.26 | Is there good color and brightness contrast between image and background colors? | X O O | 02/May/2010 |
| 7.27 | Have light, bright, saturated colors been used to emphasize data and have darker, duller, and desaturated colors been used to de-emphasize data? | O X O | 02/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 7.28 | Is the first word of each menu choice the most important? | X O O | 02/May/2010 |
| 7.29 | Does the system provide *mapping:* that is, are the relationships between controls and actions apparent to the user? | X O O | 02/May/2010 |
| 7.30 | Are input data codes distinctive? | X O O | 02/May/2010 |
| 7.31 | Have frequently confused data pairs been eliminated whenever possible? | X O O | 02/May/2010 |
| 7.32 | Have large strings of numbers or letters been broken into chunks? | O O X | 02/May/2010 |
| 7.33 | Are inactive menu items grayed out or omitted? | X O O | 02/May/2010 |
| 7.34 | Are there menu selection defaults? | X O O | 02/May/2010 |
| 7.35 | If the system has many menu levels or complex menu levels, do users have access to an on-line spatial menu map? | O O X | 02/May/2010 |
| 7.36 | Do GUI menus offer affordance: that is, make obvious where selection is possible? | X O O | 02/May/2010 |
| 7.37 | Are there salient visual cues to identify the active window? | O O X | 02/May/2010 |
| 7.38 | Are function keys arranged in logical groups? | O O X | 02/May/2010 |
| 7.39 | Do data entry screens and dialog boxes indicate when fields are optional? | X O O | 02/May/2010 |
| 7.40 | On data entry screens and dialog boxes, are dependent fields displayed only when necessary? | O O X | 02/May/2010 |

### 8. Fexibility and Minimalist Design

Accelerators-unseen by the novice user-may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. Provide alternative means of access and operation for users who differ from the “average” user (e.g., physical or cognitive ability, culture, language, etc.)

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 8.1 | If the system supports both novice and expert users, are multiple levels of error message detail available? | O O X | 02/May/2010 |
| 8.2 | Does the system allow novices to use a keyword grammar and experts to use a positional grammar? | O O X | 02/May/2010 |
| 8.3 | Can users define their own synonyms for commands? | O O X | 02/May/2010 |
| 8.4 | Does the system allow novice users to enter the simplest, most common form of each command, and allow expert users to add parameters? | O O X | 02/May/2010 |
| 8.5 | Do expert users have the option of entering multiple commands in a single string? | O O X | 02/May/2010 |
| 8.6 | Does the system provide function keys for high-frequency commands? | O X O | 02/May/2010 |
| 8.7 | For data entry screens with many fields or in which source documents may be incomplete, can users save a partially filled screen? | X O O | 02/May/2010 |
| 8.8 | Does the system automatically enter leading zeros? | O X O | 02/May/2010 |
| 8.9 | If menu lists are short (seven items or fewer), can users select an item by moving the cursor? | O O X | 02/May/2010 |
| 8.10 | If the system uses a type-ahead strategy, do the menu items have mnemonic codes? | O O X | 02/May/2010 |
| 8.11 | If the system uses a pointing device, do users have the option of either clicking on fields or using a keyboard shortcut? | O X O | 02/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 8.12 | Does the system offer "find next" and "find previous" shortcuts for database searches? | O X O | 02/May/2010 |
| 8.13 | On data entry screens, do users have the option of either clicking directly on a field or using a keyboard shortcut? | X O O | 02/May/2010 |
| 8.14 | On menus, do users have the option of either clicking directly on a menu item or using a keyboard shortcut? | X O O | 02/May/2010 |
| 8.15 | In dialog boxes, do users have the option of either clicking directly on a dialog box option or using a keyboard shortcut? | X O O | 02/May/2010 |
| 8.16 | Can expert users bypass nested dialog boxes with either type-ahead, user-defined macros, or keyboard shortcuts? | O X O | 02/May/2010 |

### 9. Aesthetic and Minimalist Design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 9.1 | Is only (and all) information essential to decision making displayed on the screen? | X O O | 02/May/2010 |
| 9.2 | Are all icons in a set visually and conceptually distinct? | X O O | 02/May/2010 |
| 9.3 | Have large objects, bold lines, and simple areas been used to distinguish icons? | X O O | 02/May/2010 |
| 9.4 | Does each icon stand out from its background? | X O O | 02/May/2010 |
| 9.5 | If the system uses a standard GUI interface where menu sequence has already been specified, do menus adhere to the specification whenever possible? | O O X | 02/May/2010 |
| 9.6 | Are meaningful groups of items separated by white space? | X O O | 02/May/2010 |
| 9.7 | Does each data entry screen have a short, simple, clear, distinctive title? | X O O | 02/May/2010 |
| 9.8 | Are field labels brief, familiar, and descriptive? | X O O | 02/May/2010 |
| 9.9 | Are prompts expressed in the affirmative, and do they use the active voice? | X O O | 02/May/2010 |
| 9.10 | Is each lower-level menu choice associated with only one higher level menu? | O X O | 02/May/2010 |
| 9.11 | Are menu titles brief, yet long enough to communicate? | X O O | 02/May/2010 |
| 9.12 | Are there pop-up or pull-down menus within data entry fields that have many, but well-defined, entry options? | X O O | 02/May/2010 |

### 10. Help and Documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 10.1 | If users are working from hard copy, are the parts of the hard copy that go on-line marked? | O O X | 02/May/2010 |
| 10.2 | Are on-line instructions visually distinct? | X O O | 02/May/2010 |
| 10.3 | Do the instructions follow the sequence of user actions? | X O O | 02/May/2010 |
| 10.4 | If menu choices are ambiguous, does the system provide additional explanatory information when an item is selected? | X O O | 02/May/2010 |
| 10.5 | Are data entry screens and dialog boxes supported by navigation and completion instructions? | X O O | 02/May/2010 |
| 10.6 | If menu items are ambiguous, does the system provide additional explanatory information when an item is selected? | X O O | 02/May/2010 |
| 10.7 | Are there memory aids for commands, either through on-line quick reference or prompting? | O O X | 02/May/2010 |
| 10.8 | Is the help function visible; for example, a key labeled HELP or a special menu? | O O X | 02/May/2010 |
| 10.9 | Is the help system interface (navigation, presentation, and conversation) consistent with the navigation, presentation, and conversation interfaces of the application it supports? | O O X | 02/May/2010 |
| 10.10 | Navigation: Is information easy to find? | X O O | 02/May/2010 |

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 10.11 | Presentation: Is the visual layout well designed? | X O O | 02/May/2010 |
| 10.12 | Conversation: Is the information accurate, complete, and understandable? | X O O | 02/May/2010 |
| 10.13 | Is the information relevant? | X O O | 02/May/2010 |
| 10.14 | Goal-oriented (What can I do with this program?) | X O O | 02/May/2010 |
| 10.15 | Descriptive (What is this thing for?) | X O O | 02/May/2010 |
| 10.16 | Procedural (How do I do this task?) | X O O | 02/May/2010 |
| 10.17 | Interpretive (Why did that happen?) | X O O | 02/May/2010 |
| 10.18 | Navigational (Where am I?) | X O O | 02/May/2010 |
| 10.19 | Is there context-sensitive help? | O X O | 02/May/2010 |
| 10.20 | Can the user change the level of detail available? | O X O | 02/May/2010 |
| 10.21 | Can users easily switch between help and their work? | O O X | 02/May/2010 |
| 10.22 | Is it easy to access and return from the help system? | O O X | 02/May/2010 |
| 10.23 | Can users resume work where they left off after accessing help? | O O X | 02/May/2010 |

### 11. Skills

The system should support, extend, supplement, or enhance the user’s skills, background knowledge, and expertise ----not replace them.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 11.1 | Can users choose between iconic and text display of information? | O X O | 03/May/2010 |
| 11.2 | Are window operations easy to learn and use? | X O O | 03/May/2010 |
| 11.3 | If users are experts, usage is frequent, or the system has a slow response time, are there fewer screens (more information per screen)? | O X O | 03/May/2010 |
| 11.4 | If users are novices, usage is infrequent, or the system has a fast response time, are there more screens (less information per screen)? | O X O | 03/May/2010 |
| 11.5 | Does the system automatically color-code items, with little or no user effort? | O X O | 03/May/2010 |
| 11.6 | If the system supports both novice and expert users, are multiple levels of detail available. | O O X | 03/May/2010 |
| 11.7 | Are users the initiators of actions rather than the responders? | X O O | 03/May/2010 |
| 11.8 | Does the system perform data translations for users? | X O O | 03/May/2010 |
| 11.9 | Do field values avoid mixing alpha and numeric characters whenever possible? | X O O | 03/May/2010 |
| 11.10 | If the system has deep (multilevel) menus, do users have the option of typing ahead? | O O X | 03/May/2010 |
| 11.12 | When the user enters a screen or dialog box, is the cursor already positioned in the field users are most likely to need? | X O O | 03/May/2010 |
| 11.13 | Can users move forward and backward within a field? | X O O | 03/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 11.14 | Is the method for moving the cursor to the next or previous field both simple and visible? | X O O | 03/May/2010 |
| 11.15 | Has auto-tabbing been avoided except when fields have fixed lengths or users are experienced? | O O X | 03/May/2010 |
| 11.16 | Do the selected input device(s) match user capabilities? | O O X | 03/May/2010 |
| 11.17 | Are cursor keys arranged in either an inverted T (best for experts) or a cross configuration (best for novices)? | X O O | 03/May/2010 |
| 11.18 | Are important keys (for example, ENTER , TAB) larger than other keys? | X O O | 03/May/2010 |
| 11.19 | Are there enough function keys to support functionality, but not so many that scanning and finding are difficult? | X O O | 03/May/2010 |
| 11.20 | Are function keys reserved for generic, high-frequency, important functions? | O O X | 03/May/2010 |
| 11.21 | Are function key assignments consistent across screens, subsystems, and related products? | O O X | 03/May/2010 |
| 11.22 | Does the system correctly anticipate and prompt for the user's probable next activity? | O O X | 03/May/2010 |

### 12. Pleasurable and Respectful Interaction with the User

The user’s interactions with the system should enhance the quality of her or his work-life. The user should be treated with respect. The design should be aesthetically pleasing- with artistic as well as functional value.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 12.1 | Is each individual icon a harmonious member of a family of icons? | X O O | 03/May/2010 |
| 12.2 | Has excessive detail in icon design been avoided? | X O O | 03/May/2010 |
| 12.3 | Has color been used with discretion? | X O O | 03/May/2010 |
| 12.4 | Has the amount of required window housekeeping been kept to a minimum? | X O O | 03/May/2010 |
| 12.5 | If users are working from hard copy, does the screen layout match the paper form? | O O X | 03/May/2010 |
| 12.6 | Has color been used specifically to draw attention, communicate organization, indicate status changes, and establish relationships? | O X O | 03/May/2010 |
| 12.7 | Can users turn off automatic color coding if necessary? | O O X | 03/May/2010 |
| 12.8 | Are typing requirements minimal for question and answer interfaces? | X O O | 03/May/2010 |
| 12.9 | Do the selected input device(s) match environmental constraints? | X O O | 03/May/2010 |
| 12.13 | If the system uses multiple input devices, has hand and eye movement between input devices been minimized? | O O X | 03/May/2010 |
| 12.14 | If the system supports graphical tasks, has an alternative pointing device been provided? | O O X | 03/May/2010 |
| 12.15 | Is the numeric keypad located to the right of the alpha key area? | X O O | 03/May/2010 |
| # | Review Checklist | Yes No N/A | Checked on |
| 12.16 | Are the most frequently used function keys in the most accessible positions? | O O X | 03/May/2010 |
| 12.17 | Does the system complete unambiguous partial input on a data entry field? | O X O | 03/May/2010 |

### 13. Privacy

The system should help the user to protect personal or private information- belonging to the user or the his/her clients.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Review Checklist | Yes No N/A | Checked on |
| 13.1 | Are protected areas completely inaccessible? | X O O | 03/May/2010 |
| 13.2 | Can protected or confidential areas be accessed with certain passwords. | X O O | 03/May/2010 |
| 13.3 | Is this feature effective and successful. | X O O | 03/May/2010 |

*References*

*Xerox Corporation, 1995. Usability Analysis & Design, Xerox Corporation.*

*Weiss, E. 1993. Making Computers People Literate. ISBN: 0-471-01877-5*

*Nielsen, J. Mack, R. 1994. Usability Inspection Methods. ISBN: 1-55542-622-0*

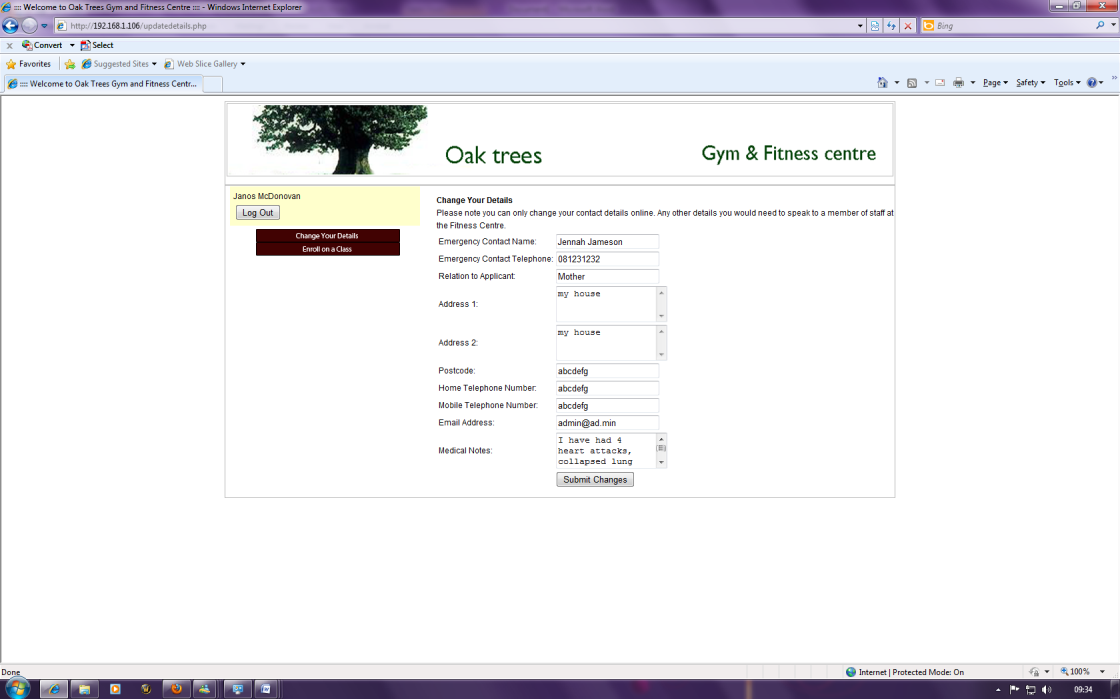
# Website Testing

Small Issues

IE: When changing details, the confirmation box comes up twice and the page almost drops part of the CSS.



The change details section is now below the menu, when it should be along side it.



So it appears there is some loss of formatting somewhere. This does not happen in the other browsers however. It only seems to be happening in IE at the moment.

### Website Test Plan - Firefox

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Content renders correctly | Loading web page into browser | Content to appear rendered correctly | Pass | 23/04/2010 |
| 2 | Links working correctly | Loading web page into browser | Links to go to the correct page | Pass | 23/04/2010 |
| 3 | Database connection established | Loading the web page and attempting to connect to db | Connects to database correctly | Pass | 23/04/2010 |
| 4 | Database information displayed correctly | Loading web page, establishing db connection | Displays information from database correctly | Pass | 23/04/2010 |
| 5 | Changes to database being permitted when required | Loading web page and change data in db | Allows changes to database when its required | Pass | 23/04/2010 |
| 6 | Logging in shows correct message if incorrect | Loading web page and attempting to log in | Should display page saying that log in details are incorrect | Pass | 23/04/2010 |
| 7 | JavaScript validating numbers in number field | Attempting to add letters to numeric field | Should error when non numeric characters are entered | Fail | 23/04/2010 |
| 8 | JavaScript validating postcode in postcode field | Attempting to enter something other than a postcode | Should error when something that is not a postcode is entered | Fail | 23/04/2010 |
| 9 | Class Booking system allows member to enroll on class | Attempting to enroll on a class | Should allow member to enroll on a class when they request it unless full | Pass | 23/04/2010 |
| 10 | Allowing the user to log out properly and closes the database connection | Attempting to log out of the system | Should show a logged out page | Pass | 23/04/2010 |

### Website Test Plan - Chrome

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Content renders correctly | Loading web page into browser | Content to appear rendered correctly | Pass | 23/04/2010 |
| 2 | Links working correctly | Loading web page into browser | Links to go to the correct page | Pass | 23/04/2010 |
| 3 | Database connection established | Loading the web page and attempting to connect to db | Connects to database correctly | Pass | 23/04/2010 |
| 4 | Database information displayed correctly | Loading web page, establishing db connection | Displays information from database correctly | Pass | 23/04/2010 |
| 5 | Changes to database being permitted when required | Loading web page and change data in db | Allows changes to database when its required | Pass | 23/04/2010 |
| 6 | Logging in shows correct message if incorrect | Loading web page and attempting to log in | Should display page saying that log in details are incorrect | Pass | 23/04/2010 |
| 7 | JavaScript validating numbers in number field | Attempting to add letters to numeric field | Should error when non numeric characters are entered | Fail | 23/04/2010 |
| 8 | JavaScript validating postcode in postcode field | Attempting to enter something other than a postcode | Should error when something that is not a postcode is entered | Fail | 23/04/2010 |
| 9 | Class Booking system allows member to enroll on class | Attempting to enroll on a class | Should allow member to enroll on a class when they request it unless full | Pass | 23/04/2010 |
| 10 | Allowing the user to log out properly and closes the database connection | Attempting to log out of the system | Should show a logged out page | Pass | 23/04/2010 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Test Data** | **Expected Result** | **Pass/Fail** | **Date of Test** |
| 1 | Content renders correctly | Loading web page into browser | Content to appear rendered correctly | Pass | 23/04/2010 |
| 2 | Links working correctly | Loading web page into browser | Links to go to the correct page | Pass | 23/04/2010 |
| 3 | Database connection established | Loading the web page and attempting to connect to db | Connects to database correctly | Pass | 23/04/2010 |
| 4 | Database information displayed correctly | Loading web page, establishing db connection | Displays information from database correctly | Pass | 23/04/2010 |
| 5 | Changes to database being permitted when required | Loading web page and change data in db | Allows changes to database when its required | Pass | 23/04/2010 |
| 6 | Logging in shows correct message if incorrect | Loading web page and attempting to log in | Should display page saying that log in details are incorrect | Pass | 23/04/2010 |
| 7 | JavaScript validating numbers in number field | Attempting to add letters to numeric field | Should error when non numeric characters are entered | Fail | 23/04/2010 |
| 8 | JavaScript validating postcode in postcode field | Attempting to enter something other than a postcode | Should error when something that is not a postcode is entered | Fail | 23/04/2010 |
| 9 | Class Booking system allows member to enroll on class | Attempting to enroll on a class | Should allow member to enroll on a class when they request it unless full | Pass | 23/04/2010 |
| 10 | Allowing the user to log out properly and closes the database connection | Attempting to log out of the system | Should show a logged out page | Pass | 23/04/2010 |

### Website Test Plan - Internet Explorer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Actual Result** | **Result (P/F)** | **Comment** | **Date of Test** |
| 1 | To check to see if the content renders correctly when the page is loaded | Pages load fine | Pass | Pages loaded fine, css working correctly | 23/04/2010 |
| 2 | Links working correctly and going to the correct page | Links go to correct places | Pass | Some of the pages need some content, but at this stage as long as the links work its good | 23/04/2010 |
| 3 | Allowing a connection to the database | Logs in a user with no problems | Pass | The fact that a user is being logged in is all I really needed to do to test this | 23/04/2010 |
| 4 | Displaying information from the database to the screen | Loaded the change details page, was populated fine | Pass | All fields loaded correctly, some differences in font but that was due to the different text boxes | 23/04/2010 |
| 5 | Allowing changes to be made to database data when required | Changes are allowed to be made to the db | Pass | The changes are being permitted through the sql queries in the php | 23/04/2010 |
| 6 | Logging in shows the correct message when an incorrect value is entered | Attempting to log in brings up incorrect message | Pass | Loads the page saying that the username and password is incorrect - which is right! | 23/04/2010 |
| 7 | JavaScript validating correctly if a non numeric character is entered in numeric field | Allowed changes to go through unvalidated | Fail | This is something that could be rather simple to implement, but at this stage we don't have the time. If it was going live however, | 23/04/2010 |
| 8 | JavaScript validating correctly if a value that isn't a postcode is entered in a postcode field | Allowed changes to go through unvalidated | Fail | Then this would be implemented to prevent changes done to the database that would affect the fields in question. | 23/04/2010 |
| 9 | Allowing a member to enroll onto a class using the class booking system | Allowed the user to book onto a class | Pass | Spelling error on the pop up box that asks for confirmation of the class booking | 23/04/2010 |
| 10 | Allowing the user to log out properly and closes the database connection | Returned the user to the home page | Pass | Ideally showing a page that would let the user know they had logged out would be better | 23/04/2010 |

### Website Test Log - Firefox

### Website Test Log – Chrome

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Actual Result** | **Result (P/F)** | **Comment** | **Date of Test** |
| 1 | To check to see if the content renders correctly when the page is loaded | Pages are displayed as per design | Pass | Everything loads correctly when the page is loaded into the browser | 23/04/2010 |
| 2 | Links working correctly and going to the correct page | Links go to correct destination | Pass | All links take user to the correct page when they are clicked on | 23/04/2010 |
| 3 | Allowing a connection to the database | User logs in without a problem | Pass | If the user couldn't log in then there would be an issue with the database | 23/04/2010 |
| 4 | Displaying information from the database to the screen | Loads the page with changes fine | Pass | Page is loaded with changes done in another browser, so they are getting saved to db | 23/04/2010 |
| 5 | Allowing changes to be made to database data when required | Allows the changes through fine | Pass | Changes are written back to the database like it should be | 23/04/2010 |
| 6 | Logging in shows the correct message when an incorrect value is entered | Shows incorrect login page | Pass | User cannot log in if they enter an incorrect username and password - works as intended | 23/04/2010 |
| 7 | JavaScript validating correctly if a non numeric character is entered in numeric field | Doesn't validate how it is supposed to, allows anything through | Fail | At this stage, the fact that a small piece of JavaScript does not work is not really a cause for concern. If this was being launched then | 23/04/2010 |
| 8 | JavaScript validating correctly if a value that isn't a postcode is entered in a postcode field | Doesn't validate how it is supposed to, allows anything through | Fail | Ideally this would need to be fixed and tested again to ensure that no illegal entries could be added to the database | 23/04/2010 |
| 9 | Allowing a member to enroll onto a class using the class booking system | Allows the user to enroll on class | Pass | Had to cancel booking to enroll on it again, as logged in using the same login details | 23/04/2010 |
| 10 | Allowing the user to log out properly and closes the database connection | Returns to home page | Pass | Should show that they have logged out, but it works, that's what is important! | 23/04/2010 |

### Website Test Log - Internet Explorer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Purpose of Test** | **Actual Result** | **Result (P/F)** | **Comment** | **Date of Test** |
| 1 | To check to see if the content renders correctly when the page is loaded | Page loads with no errors | Pass | Page renders fine, no errors or anything when it loads | 23/04/2010 |
| 2 | Links working correctly and going to the correct page | Links go to correct pages | Pass | The links work correctly, taking the user to the correct page | 23/04/2010 |
| 3 | Allowing a connection to the database | Logs the user in with no problems | Pass | Seeing as the user can log in, there are no problems with the database connection | 23/04/2010 |
| 4 | Displaying information from the database to the screen | All details from previous browser test loaded | Pass | As details from the previous browser test are loading, it is a sign that this is still working across different platforms | 23/04/2010 |
| 5 | Allowing changes to be made to database data when required | Details are submitted, page breaks afterwards | Pass | Changes are submitted but page layout goes funny afterwards... hmm. Dialog box appears twice and then the page breaks | 23/04/2010 |
| 6 | Logging in shows the correct message when an incorrect value is entered | Shows error if wrong details | Pass | Loads the page saying that the username and password is incorrect - which is right! | 23/04/2010 |
| 7 | JavaScript validating correctly if a non numeric character is entered in numeric field | Not bringing up an error if an incorrect value is entered | Fail | This is something that could be rather simple to implement, but at this stage we don't have the time. If it was going live however, | 23/04/2010 |
| 8 | JavaScript validating correctly if a value that isn't a postcode is entered in a postcode field | Not bringing up an error if an incorrect value is entered | Fail | Then this would be implemented to prevent changes done to the database that would affect the fields in question. | 23/04/2010 |
| 9 | Allowing a member to enroll onto a class using the class booking system | Allows user to enroll on a class | Pass | Enrolls fine and doesn't break the page after the dialog box has been and gone | 23/04/2010 |
| 10 | Allowing the user to log out properly and closes the database connection | Logs the user out | Pass | User is successfully logged out of the system when clicking logout. | 23/04/2010 |

### Overview

While testing the website, I considered testing it on the web server I have set up, or testing it through another machine to get the feel of it from a client perspective. After disabling the firewall of the web server, I was able to make contact with the server from my client machine and test each page as if it were from a users point of view. Choosing to use three web browsers could be one of the most accurate methods of testing a website, simply because not everyone uses the same browser. While I have only tested on the three main ones, the test results infer that if it was accessed on something other than Internet Explorer, Google Chrome or Mozilla Firefox, they should have very few problems.

The testing of the website proved to be successful in that the majority of the tests gave a pass result. The only tests that failed on each of the browser tests was the JavaScript validation, which would prevent a user entering a non numeric character into a field that required characters, such as telephone number and a series of characters into the postcode field that didn't correspond to something that resembled a postcode.

Whilst the testing was mostly successful, a few small issues were highlighted. As shown above the testing tables, when trying to change details using Internet Explorer, the dialog box confirming the changes have been made appears twice, and after you have clicked ok to get rid of it, the page layout changes. This does not happen on the class booking page, despite the fact a similar dialog box is used. This is only happening with Internet Explorer, and only on this page.

One thing that does stand out is the current lack of content on some of the pages. Whilst this web based addition to our main C# application will never get published, it would be nice if it was something that resembled a finished product. However due to the time constraints, this is not going to be possible this late in the project. The pages themselves resemble the design closely, and I feel that in its development, the design has been followed thoroughly to make it as true to what was required as possible.

1. http://www1.euro.dell.com/uk/en/business/Servers/poweredge-t110/pd.aspx?refid=poweredge-t110&s=bsd&cs=ukbsdt1 [↑](#footnote-ref-1)
2. http://www1.euro.dell.com/uk/en/business/Desktops/vostro-230st/pd.aspx?refid=vostro-230st&s=bsd&cs=ukbsdt1 [↑](#footnote-ref-2)
3. http://www.cisco.com/en/US/products/hw/routers/ps380/index.html [↑](#footnote-ref-3)
4. http://www.cisco.com/en/US/products/ps6545/index.html [↑](#footnote-ref-4)
5. http://www.ciscosystems.com/en/US/prod/collateral/switches/ps5718/ps708/product\_solution\_overview0900aecd80630c78.html [↑](#footnote-ref-5)
6. Topology created using Cisco Packet Tracer (http://www.cisco.com/web/learning/netacad/course\_catalog/PacketTracer.html) [↑](#footnote-ref-6)