

Gym Membership Training System

Prepared for: Colin Manning

Prepared by: Michael Mahoney, John Vincent O Dwyer, Robert Gabriel, Piotr Kawalec

9 November 2014

Group Project : 2

Github : https://github.com/RobertJGabriel/Third-Year-Group-Project

Presentation flow

Preconditions:

#1 have admin account and three trainers ready plus a basic member as well.

#2 all tables cleared except “members”

#3 browser cleaned

Task 1: (student)

A user tries to register and login into the application then after he wants to:

1. see his settings and update his photo
2. book a training session
   1. use card view on the profile page
   2. use ‘Book’ menu option and then a calendar
3. view his bookings
4. input his workout

Task 2: (trainer)

User registers to the system and then gets promoted to trainer then after:

1. user logs in to the system.
2. ….

Task 3: (admin)

User logs in as an admin then he is able to:

1. manage the admin area
   1. view user’s details
   2. update user’s details
   3. change the status of the user [123]
   4. delete user from the system
2. schedule a working day for the trainer (or delete if already exists)
3. view trainer’s schedules/bookings for the day in the “Book” menu option
4. + standard user activities

Index

**Part 1 - Specification, Analysis, Research**

# Glossary of terms

# Assumptions

# Project Drivers

# The purpose of project

# The Goals

# The Stakeholders

# Project Constraints

# Mandated Constraints

# Solution Constraints

# Implementation Environment Constrains

# Partner or Collaborative Applications

# Schedule Constraints

# Budget Constraints

# Naming Conventions and Definitions

# Relevant Facts and assumptions

# Business Rules

# Functional Requirements

# The scope of work

# Context Diagram

* + 1. Work partitioning
    2. Business Use Cases
  1. Business Data Model
  2. The Scope of the Product
  3. Functional Requirements

# Non-functional Requirements

* 1. Appearance and style
  2. Usability and Humanity
  3. Performance
  4. Maintainability and support
  5. Security
  6. Usability and responsive design.

**Part 2 - Implementation and Testing**

1. Database design
2. Modeling and prototyping
   1. User Interface
   2. Mobile design
3. Code implementation
   1. HTML and CSS
   2. PHP and SQL
4. External code implementation (APIs)
5. Code testing

**Part 3 - Project Management and Communication**

1. Initial assessment
2. Work organisation
3. Project progress
4. Project issues and solutions
5. Learning outcome and final conclusions

# 

Part 1- **Specification, Analysis, Research**

# Glossary of Terms

Dead Lift : a lift made from a standing position, without the use of a bench or other equipment.

Cardio : cardiovascular exercise.

Personal Trainer : A personal trainer is a fitness professional involved in exercise prescription and instruction. They motivate clients by setting goals and providing feedback and accountability to clients. Trainers also measure their client's strengths and weaknesses with fitness assessments.

Administration : the process or activity of running a business, organization, etc.

"the day-to-day administration of the company"

Gym : A Gymnasium.

Heart rate, or heart pulse: is the speed of the heartbeat measured by the number of heartbeats per unit of time — typically beats per minute (bpm).

Body Mass Index (BMI): is a simple index of weight-for-height that is commonly used to classify underweight, overweight and obesity in adults. It is defined as the weight in kilograms divided by the square of the height in metres (kg/m2).

Member: a person that is registered with Gym Training System and can book a session with the trainer.

Super Admin: (gym manager) a person that is responsible for managing all aspects of the business

Student: a person who is studying at a university or other place of higher education.

Schedule: period of time specified for a particular day and the trainer that allows the members to book a training session within it.

Trainer Availability: an Trainer’s work availability may be a set number of days and hours for when a gym member can book them. For example, a schedule could be Monday - Friday, from 9 am or 5 pm.

Workout: a session of vigorous physical exercise or training. Within the application it is recognized as recorded values for particular type of exercise.

Cardio workout: a session of motion exercise, where measured attributes are distance and time.

Booking (training session): appointment between the trainer and the member for a particular day and hour.

API: Application Programming Interface - provided from external sources set of methods that allow for communication between specified applications and services.

CIT Id, member Id: a number that identifies person registered in CIT. It consists of the letter ‘R’ and followed by 8 digits.

# Assumptions

Persons registered in CIT have the same structure of the Id numbers.

Application design intention is for CIT gym, with eventual possibility of adaptation to other locations.

# Other assumptions are stated during the document.

# Project Drivers

## The purpose of project

## Create an application for the members of gym for booking training sessions, to track their progress over time.

## The Goals

The application should work equally well on desktop and mobile devices.

The application will be sold as an off-the-shelf application which can be personalised for the Gym.

The application will allow for:

* sign up/log in and profile maintenance operations
* booking session/appointment with a trainer the date/time they are available
* place input of a gym session.

Trainers will be able to provide date/times they are available

Details of Trainers, location, branding etc. can be edited through an administration account.

New accounts for gym trainers will be created using the administration account.

When the gym member books a training time, it should map a root from their home address to the gyms using google maps.

Application will be able to pull different gym photos from flickr for the background images.

The app should include a news feed from a fitness website /twitter account.

The app design should take into account the audience for that website.

Book a training session (Input training time event to google calendar.)

## 

## 

## The Stakeholders

**Gym manager/Super Admin:**

This is a person who is responsible for managing everyday aspects of the business as well as a whole structure. He will be in control of any functional detail of working application and also have final word on acceptance of delivered application.

**Trainers:**

A personal trainer is a fitness professional involved in [exercise prescription](http://en.wikipedia.org/wiki/Exercise_prescription) and instruction. They motivate clients by setting goals and providing feedback and accountability to clients. Trainers also measure their client's strengths and weaknesses with fitness assessments. These fitness assessments may also be performed before and after an exercise program to measure their client's improvements in [physical fitness](http://en.wikipedia.org/wiki/Physical_fitness).

**The Gym member:**

A Gym Member is anyone who is a current student of Cit. That uses and has access to the gym. They can book training sessions with the trainers.

**System designers/developers:**

Group of persons involved in design and implementation of:

* User Interfaces
* Backend code
* Database structure
* Testing

The individuals need to possess knowledge of web app usability evaluation, user experience aspects, technologies used to develop interaction between user and background mechanisms and also be able to create data structures that support required functionality. Additionally ability of test performing will be included. Any conflicts or major solution issues will be addressed during ad-hoc meetings of stakeholders involved.

**Project supervisor:**

A person designated to provide intellectual supervision over all of the project’s stages.

# 

# 

# Project Constraints

## Mandated Constraints

### Solution Constraints

The application must be able to work the same in firefox and chrome browsers.

These are major players of the market and will serve most critical amount of users.

Multiple devices with different screen sizes must be fully supported.

Users will connect to the application via wide range of devices.

It should not limit the usage of the application.

### Implementation Environment Constrains

Any thing we need to keep with about

The application must be implemented to use of selected database solution.

Database will facilitate use of storage and also have positive impact on performance.

### Partner or Collaborative Applications

The application will utilize third party systems. It will connect to the APIs of Google Maps, Google Calendar, Flicker in order to provide enhanced information. The operational details for each remote system will need to be set.

### Schedule Constraints

How fast the system is and how quickly we need to get it done ( The 12 week)

Time table goes here.

The time for project completion is 11th of December 2014

### Budget Constraints

Free because where poor students

### Areas for growth:

* allow for non student members
* record additional training details
* Integrate the login system

## Relevant Facts and assumptions

There is no current system

### Business Rules

1. Student members must have valid ID number
2. Email address and password must be provided to access system
3. Booking can be done for maximum 30 days in advance
4. Booking cancellation is allowed at any time

### 

### Assumptions

* That the gym is open to only cit students and staff

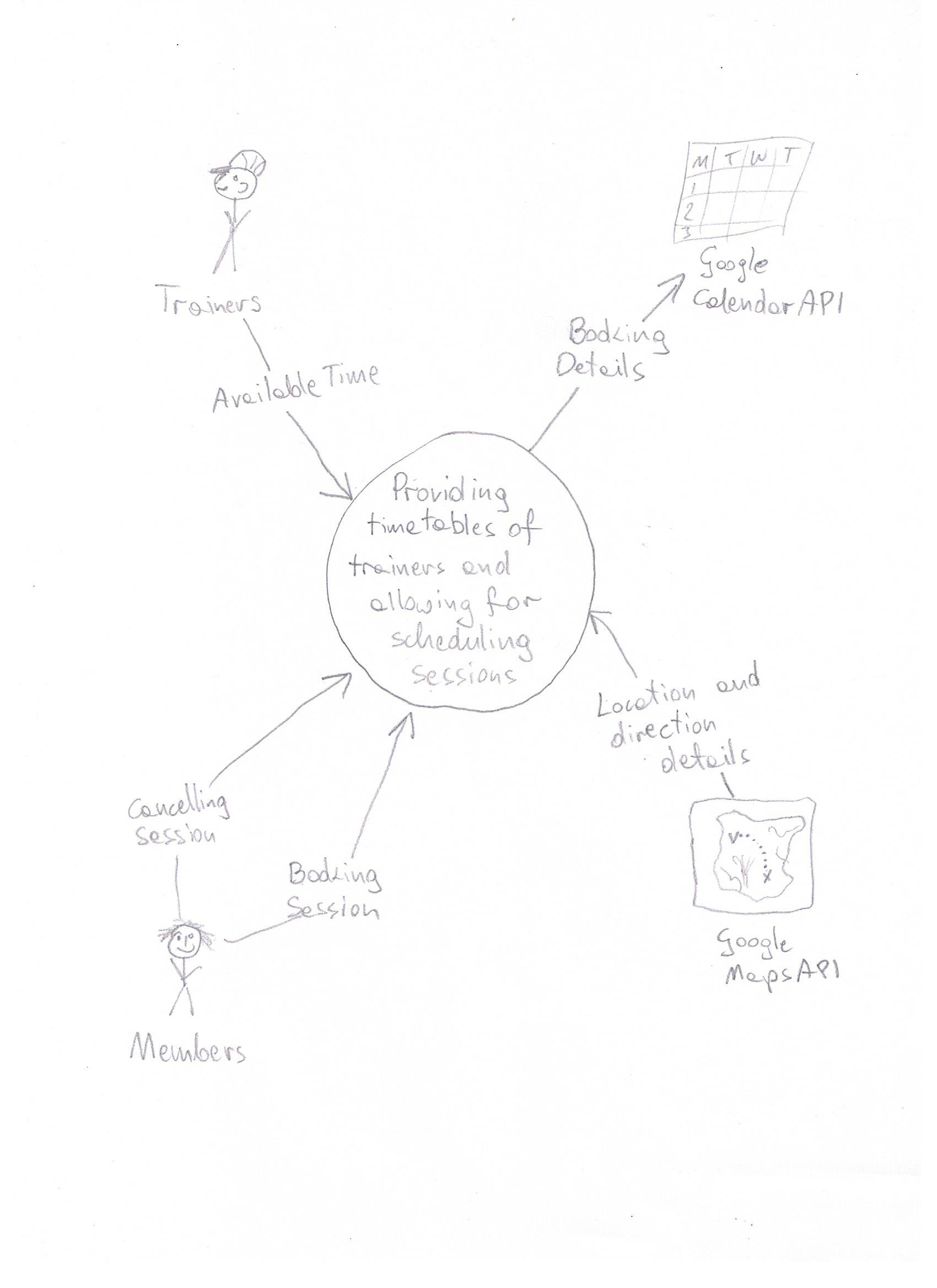
# Functional Requirements

## The Scope of the Work

This application will automate existing business processes for scheduling training sessions. It will provide self contained remote access to procedures necessary for making appointments between trainers and trainees. It will integrate external sources to enhance user experience.

## Context Diagram

## 



## 

## 

## 

## Work Partitioning

We as a team decided to undertake a task of building our own mini framework, so that the code can be reused as an original framework from gyms in the college system even beyond cit.

A framework gives you standard solutions to typical problems, e.g. for an online shop that can cover the functionality for a customer login (including session handling), a shopping cart, placing orders.

The big advantage of using a framework is that

1. You don't need to reinvent the wheel, the code is already there
2. The code (usually) works, it is already tested
3. Specifically for user authentication, you will most probably have fewer security leaks as if you invented something from scratch yourself

The big disadvantage is that

1. If you want to extend the functionality, you have to understand OPC (*other peoples code*)
2. If the framework contains a security hole and an exploit is available, your site is immediately vulnerable, but you may not have the knowledge to fix it yourself. So you need to keep a constant lookout on security bulletins, updates, fixes etc.

## Business Use Cases

### Typical business events:

**Student should be able to**

* Sign up.
* Login.
* Recover their password.
* Edit profile.
* Edit settings.
* View Gym Trainers contact details.
* View appointments for any day for any trainer.
* Book appointments for training with the trainer.
* Cancel own bookings.
* Input results from gym session (weights, miles run etc).
* View results from gym sessions.

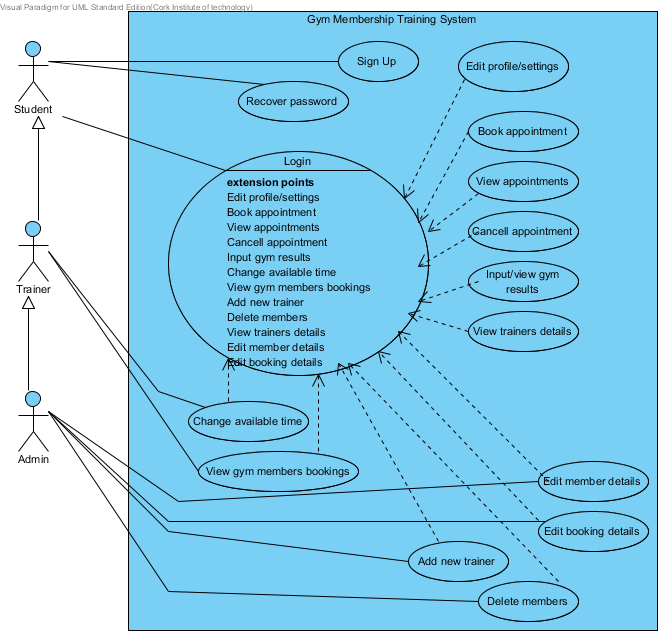
**Trainer should be able to do above and also**

* Cancel any bookings
* View gym members who booked training times

**Administrator should be able to do above and also**

* Add a new trainer
* Edit/view all members details
* Edit/view all bookings details
* Delete members
* Set/change his available times to train at for trainers

### Use case diagram:



### Use Cases:

|  |  |
| --- | --- |
| Use Case #1 | **Sign up** |
| Actor | Any person registered in CIT |
| Precondition | Has valid CIT ID number |
| Postcondition | Person is now a member of the gym booking website |
| Main Path | 1. Person inputs required information 2. System verifies information 3. System creates user account 4. System loads up login page |
| Alternative Path 1 | @2. System verifies user already exists  System displays message and allows user to reenter details or  switch to log in page @2. User input is incomplete  System displays message and allows user to continue |
| Exception | @2. System rejects user sign-up if user does not provide valid Id number. |

|  |  |
| --- | --- |
| Use Case #2 | **Login** |
| Actor | Member |
| Precondition | Must be a member |
| Postcondition | They are now logged in |
| Main Path | 1. User inputs his email address and password into login form. 2. System checks if they match 3. System brings them to the Profile page. |
| Alternative Path 1 | @2. Users inputs incorrect email address or password details  System display that the user information is incorrect,  and allows for reentering information |
| Alternative Path 2 | @2. Users inputs not existing memberId  System alerts user that there is no account registered |

|  |  |
| --- | --- |
| Use Case #3 | **Recover password** |
| Actor | Member |
| Precondition | Is a registered member. |
| Postcondition | New password is set. |
| Main Path | 1. User selects ‘recover password’ 2. System prompts for email details 3. User enter email address 4. System sends new password on the email account |
| Alternative Path 1 | @4. Email address does not exists  System informs user about error and allows for re-enter the email |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #4 | **Input/view gym results info** |
| Actor | Member |
| Precondition | Is a registered member. |
| Postcondition | New weight lifting info has been added. Database updated. |
| Main Path | 1. User selects ‘Workout info’. 2. System brings up ‘Workout info’ page. 3. User selects a workout session. 4. System brings up a form. 5. User inputs sets amount into ‘Sets’ field. 6. User inputs reps amount into ‘Reps’ field. 7. User inputs weight amount into ‘Weight’ field. 8. User selects ‘Submit’. 9. System adds info to database. 10. System updates ‘Workout info’ page. |
| Alternative Path 1 | @8. User leaves a field blank and selects ‘Submit’.  System highlights the blank field. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #5 | **Book appointment** |
| Actor | Member |
| Precondition | Is signed in, Is a member |
| Postcondition | Appointment is scheduled |
| Main Path | 1. User selects the day for booking 2. System displays available trainers and their time schedule 3. User selects hour 4. System confirms appointment |
| Alternative Path 1 | @3. User selects invalid time  System prompts user for selecting different time |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #6 | **Input available times.** |
| Actor | Trainer |
| Precondition | Is a registered member. |
| Postcondition | Database is updated with the available times. |
| Main Path | 1. User selects ‘Update Available Times’. 2. System brings up a form. 3. User selects a date. 4. User selects hours available. 5. User selects ‘Submit’. 6. System adds info to database. 7. System updates calendar. |
| Alternative Path 1 | 1. User doesn’t select a date/hours. 2. System highlights date/hours field. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #7 | **Change profile settings.** |
| Actor | Member |
| Precondition | Is a registered member. |
| Postcondition | Database is updated with preferred profile settings. |
| Main Path | 1. User selects ‘Settings’. 2. System brings up settings page. 3. User selects prefered settings. 4. System changes profile look. |
| Alternative Path 1 |  |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #8 | **Upload background photo.** |
| Actor | Member |
| Precondition | Is a registered member. |
| Postcondition | Database is updated with background photo. |
| Main Path | 1. User selects ‘Settings’. 2. System brings up settings page. 3. User selects ‘Choose File’. 4. System brings up file chooser window. 5. User selects a file from their system. 6. System updates the inner html with file name. 7. User selects ‘Upload Photo’. 8. System changes background photo. |
| Alternative Path 1 |  |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #9 | **View bookings for any day for any trainer** |
| Actor | Member |
| Precondition | Is a registered member. |
| Postcondition | Schedule is displayed |
| Main Path | 1. User enters details for interested day 2. System displays schedule information |
| Alternative Path 1 |  |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #10 | **Cancel booking** |
| Actor | Member |
| Precondition | Is a registered member |
| Postcondition | Appointment cancelled |
| Main Path | 1. User selects cancel appointment option 2. System prompts for appointment details 3. User enters appointment details 4. System confirms cancellation |
| Alternative Path 1 | @4. Appointment details are incorrect  System displays information and allows for re-entering |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #11 | **Delete members** |
| Actor | Trainer (for students), Admin (for students and trainers) |
| Precondition | Is a registered member |
| Postcondition | Member account deleted |
| Main Path | 1. User selects delete member option 2. System prompts for account details 3. User provides information 4. System checks for account existence and asks for deleting confirmation 5. User confirms |
| Alternative Path 1 | @4. User has no privileges for operation  System displays error message |
| Alternative Path 2 | @4. User account does not exists  System displays error message |

|  |  |
| --- | --- |
| Use Case #12 | **Add new trainer** |
| Actor | Admin |
| Precondition | User has admin rights |
| Postcondition | Trainer account created |
| Main Path | 1. User “selects add new trainer” 2. System prompts for account details 3. User enters details 4. System verifies information and displays confirmation |
| Alternative Path 1 | @4. Entered details are incomplete  System displays information and allows for re-entering |
| Alternative Path 2 | @4. Entered details already exist  System displays information and prompts for new entry |

|  |  |
| --- | --- |
| Use Case #13 | **Edit/view all members details** |
| Actor | Admin |
| Precondition | User is logged in and has admin rights |
| Postcondition | Users information is updated |
| Main Path | 1. User goes to account settings 2. User selects update information 3. System searches for and returns users current information 4. User inputs new updates 5. System asks for confirmation 6. User confirms 7. System writes new updates to database |
| Alternative Path 1 | @4. User inputs invalid information  System informs user of invalid input |
| Alternative Path 2 |  |

## 

|  |  |
| --- | --- |
| Use Case #14 | **Edit/view all booking detail** |
| Actor | Admin |
| Precondition | User is logged in and has admin rights |
| Postcondition | Booking information is updated |
| Main Path | 1. User goes to account settings 2. User selects update information 3. System searches for and returns users current information 4. User inputs new updates 5. System asks for confirmation 6. User confirms 7. System writes new updates to database |
| Alternative Path 1 | @4. User inputs invalid information  System informs user of invalid input |
| Alternative Path 2 |  |

## Data Model

## 

## 

## The Scope of the Product

## 

Scope of product identifies boundaries between users and the product. It separates system automated processing from activities that must be provided by the user.

|  |  |
| --- | --- |
| Actor | System |
| Provide personal details | Store and maintain the information |
| Provide availability time | Store information and display in schedule timetable |
| Provide booking details | Store information and update appropriate schedule |
| Manage stored information | Provide managing capability for the stored information |

## Functional Requirements

Requirement # 1 Use Case: 1, 2

**Description:** The application should record member details.

**Rationale:** To be able to login and manage information about gym activities

**Originator:**  Gym business owner

**Fit Criteria:**  Created account should be instantly ready to use

**Dependencies:** None

**Priority:**  Very high

**History:**  Created: 10/10/2014

Requirement # 2 Use Case: 3

**Description:** Member should be able to recover password

**Rationale:** If not, the member will not be able to manage the account. Forgetting passwords is quite common.

**Originator:** Gym business owner, members

**Fit Criteria:** This feature should be automated.

**Dependencies:** Req#1

**Priority:** High

**History:** Created: 10/10/2014

Requirement # 3 Use Case: 4

**Description:** Member should be able to record/view his gym session details.

**Rationale:** This has a goal of recording and tracking member’s progress and allowing to make training decisions based on the history of previous sessions

**Originator:** Members

**Fit Criteria:** The results shall be displayed in time progress chart

**Dependencies:** Req#1

**Priority:** Low

**History:**  Created: 10/10/2014

Requirement # 4 Use Case: 5

**Description:** Member should be able to book a session with the trainer of his choice.

**Rationale:** Allowing for scheduling appointments is the main functionality of the application.

**Originator:** Gym business owner.

**Fit Criteria:** The allowed booking times should be presented to member and after selection be immediately recorded to trainer and member schedule system.

**Dependencies:** Req#1, #5

**Priority:** Very high

**History:** Created: 11/10/2014

Requirement # 5 Use Case: 6

**Description:** Trainer should be able to enter day and time he is available for members to book a training session.

**Rationale:** This is the core part on application functionality.

**Originator:** Gym business owner.

**Fit Criteria:** After submitting the details of trainer’s availability, the recorded day and time should be listed in booking list.

**Dependencies:** Req#1

**Priority:** Very high

**History:** Created: 10/10/2014

Requirement #6 Use Case: 7

**Description:** Member should be able to change his profile details/settings.

**Rationale:** This is part of customization provided for members to ensure individual preferences and also to take into account any change in the member’s recorded personal details.

**Originator:** Common stakeholders agreement.

**Fit Criteria:** This brings a value from usability point of view.

**Dependencies:** Req#1

**Priority:** Medium

**History:** Created: 11/10/2014

Requirement #7 Use Case: 8

**Description:** The application should allow the member to upload his profile photo.

**Rationale:** This will provide interface enhancements for the users.

**Originator:** Members.

**Fit Criteria:** The photo should be available in place where member’s details are displayed.

**Dependencies:** Req#1

**Priority:** Medium.

**History:** Created: 11/10/2014

Requirement #8 Use Case: 9

**Description:** The application should allow a member to view all bookings for every trainer.

**Rationale:** This part of functionality will give the user information necessary for his own booking decision.

**Originator:** Members

**Fit Criteria:** The view should reflect actual state of bookings.

**Dependencies:** Req#1, 5

**Priority:** High

**History:** Created: 10/10/2014

Requirement #9 Use Case: 10

**Description:** Member should be able to cancel his booking for a session.

**Rationale:** To be able to update application records according to real events.

**Originator:** Stakeholders

**Fit Criteria:** Cancelling is a part of response process for individual happenings, the cancelled time should be instantly available for other booking.

**Dependencies:** Req#1, 4, 5

**Priority:** High

**History:** Created: 11/10/2014

Requirement #10 Use Case: 11

**Description:** The application should allow for deleting members accounts.

**Rationale:** Members that no longer need or want to hold account in the Gym Booking system.

**Originator:** Members.

**Fit Criteria:** Accounts that are deleted should not leave any remaining details related to them.

**Dependencies:** Req#1

**Priority:** High.

**History:** Created: 10/10/2014

Requirement #11 Use Case: 12

**Description:** The application should allow the admin to add trainers as a special type of member.

**Rationale:** Trainer is a member that will need additional abilities for scheduling his time for booking sessions.

**Originator:** Gym business owner

**Fit Criteria:** Trainer must be added to the application before he can schedule his time for session bookings.

**Dependencies:** Req#1

**Priority:** Very high

**History:** Created: 10/10/2014

Requirement # 12 Use Case: 13

**Description:** The application should allow for viewing all members details by admin user.

**Rationale:** As an admin, he must be able to have an access to all members account’s details.

**Originator:** Gym business owner.

**Fit Criteria:** The application will have to identify the admin user when he logs in and display appropriate menu options.

**Dependencies:** Req#1

**Priority:** High

**History:** Created: 11/10/2014

Requirement # 13 Use Case: 14

**Description:** The trainers and admin user should be able to edit all booking details.

**Rationale:** In the case of the member not being able to access application the trainer or admin must be able to update the booking details.

**Originator:** Gym business owner.

**Fit Criteria:** The member must contact the trainer or admin using other resources in order to provide the details.

**Dependencies:** Req#1,4, 5

**Priority:** High.

**History:** Created: 10/10/2014

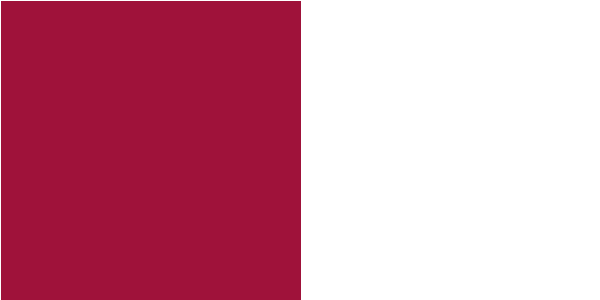
# Non-functional Requirements

## Appearance

We wanted the design of this site to be familiar. Familiarity is important in a website as it allows the user to feel comfortable with the layout and navigation from the very beginning. It’s a profile-based website so it’s important that it’s similar to other profile-based websites to keep it familiar. This site is for the students of CIT so it’s also important that it’s similar to other CIT sites in terms of look and feel.

The colour palette we chose was inspired by CIT and Cork’s signature colours: red and white.

#9f123a



**Responsiveness**

It’s important that our website is responsive in order to accommodate users who are viewing it on mobiles and tablets. For the responsive design it was important to take into account how the decrease in screen size will affect the layout of the elements. We did this by having two css files, one for mobile and one for desktops. Then with the use of media queries , it changes the css file based on the scenes with width and height.

There was problems when we had everything within the same css file. Seperating the css files allowed for two members of the team to work on the code at the same without the problem of overlapping what the others were doing.

We didn't use a css framework because we wanted a tough task. As our supervisor said about it , if you cannot do anything special dont use it .

## Style

The application will appear to the user with clean and spacious interface. The menu selection will be placed on the left hand side of the screen and the top part containing header with logo. The header and menu should be visible at all time excluding mobile version where form of button will be presented. The layout should be supplemented with the colour scheme defined in the appearance requirement.

## Ease of use

The goal of the application is to allow the user to perform his main tasks and activities within 4 clicks. The navigational labels should be self explanatory and guide the user through each level of the website. It should also provide accurate feedback to make him confident that application works as expected. When it comes to error handling, it should stop the user from continuing if any occurs by giving proper feedback along with the choice of alternate route.

## Personalization and Internationalization

The application shall allow the user to customize basic appearance like colour scheme.

It has also potential for expansion towards use in other countries by implementing range of different languages.

## Learning

The application should be ready to be used by gym members who have very little technical knowledge of using internet. Any user should be able to use the core functionality of the application within few minutes and without prior training provided.

## Accessibility

Application future growth would include support for different forms of disability aspects (visual, aural aids).

## Speed and latency (Performance)

The interface shall have maximum response time of 1 second. Within that time user must be presented with the result of his actions or its confirmation.

## Reliability and availability

The application shall be available online 24 hours/day and 365 days/year. Short period of time might be scheduled for maintenance with prior notice given.

## Capacity

The application should be able to serve 100 simultaneous users. The volume of database should be able to accommodate 1000000 different records per table.

## Maintainability and support

Working application should take no to little need for support. Initial setup expects for the database to be created and the admin account set. Post deployment maintainability could provide assistance for any emergency situations or new version release.

## Security

Security is important in any college or product that is used in the college environment , as it might and will have personal data store. So we used the following security to be included.

## The uses of mysql\_real\_escape\_string , is used with in the php coding to improve the sql and to stop against sql injections.This function must always (with few exceptions) be used to make data safe before sending a query to MySQL. This is used in forms/Inputs towards the database.

An example of an sql injection is seen below

<?php

// We didn't check $\_POST['password'], it could be anything the user wanted! For example:

$\_POST['username'] = 'aidan';

$\_POST['password'] = "' OR ''='";

// Query database to check if there are any matching users

$query = "SELECT \* FROM users WHERE user='{$\_POST['username']}' AND password='{$\_POST['password']}'";

mysql\_query($query);

// This means the query sent to MySQL would be:

echo $query;

?>

The code above allows the user to log in without a password, This is what the string echos SELECT \* FROM users WHERE user='aidan' AND password='' OR ''=''

Below is an example of the use of mysql\_real\_escape\_string() to prevent the sql injections.

var user = mysql\_real\_escape\_string( $\_POST['username'] );

var password = mysql\_real\_escape\_string($\_POST['password']) ;

Storing hashes instead of plain-text passwords assures that an attacker cannot easily recover the passwords if he gains access to the password files. Adding salt to hashed files makes hashes much harder to break with dictionary attacks. Hashing should be applied multiple times, because modern computers can perform hashing operations very quickly.

Note this if for the use of the members table.

Perform the following actions to hash and salt stored passwords:

1. **Generate a random salt value.** Each bit of salt doubles the memory and computational requirements for dictionary attacks. Recommended salt lengths are 64-bit or more. Use a random number generator of your choice to generate a random number of required size. Use this number as the salt. The following example creates a salt 64 characters (256-bit) long:

**$salt = hash('sha256', uniqid(mt\_rand(), true) . $username);**

1. **Concatenate the password and the salt.** Use string functions to concatenate the password and the salt. Make sure your buffer is large enough to store the concatenated string to prevent buffer overflows. For example:

**$storedHash = $salt . $password;**

1. **Hash the password and the salt.** Use a strong cryptographic hashing algorithm, such as SHA256, to hash the concatenated password and salt. For example:

**for ( $i = 0; $i < 50000; $i ++ )**

**{**

**$storedHash = hash('sha256', $storedHash);**

**}**

1. **Store the hash and the salt.** Store the hash in the file or database of your choosing. It is important to store both the hash and the salt, because the salt will have to be used when comparing user entered passwords to the hashed value.
2. **Implement password verification.** During authentication, the password entered by the user should be checked against the hashed and salted value. To do this, retrieve the hash value and the salt from password storage. Concatenate user input with the stored salt, hash the resulting string and compare the resulting string against the stored hash. If the resulting hash is the same as the stored hash, the user has entered the correct password. In the following example, *$userPassword* is the user enter password that needs to be validated and *$storedHash* is the stored salted hash:

**$salt = substr($storedHash, 0, 64);**

**$validateHash = $salt . $userPassword;**

**for ( $i = 0; $i < 50000; $i ++ )**

**{**

**$validateHash = hash('sha256', $validateHash);**

**}**

**$validateHash = $salt . $validateHash;**

**if ($storedHash == $validateHash)**

**{**

**//The entered password is correct.**

**}**

**else**

**{**

**//The entered password is incorrect.**

**}**

# **5. Project Issues**

# 

The problems we ran into were great and far between , from little fixes. To more troublesome problems of the timetable which is the focus on our project.

The problem we ran into can to the fact ...

Responsive design -

Displaying

# Tasks:

1. Initial planning
2. UI prototyping
3. Implementing HTML structure
4. Applying CSS
5. Applying JS
6. the
7. features
8. adding
9. DB tables planning
10. PHP classes
11. SQL
12. Server side logic
13. Connecting with UI + testing

Part 2 **- Implementation and Testing**

# 

# Database design:

For the purpose of this project we came up with the database tables’ design that will allow developed application to store and maintain information about the users and their activities. In addition it will also provide support for user’s (members) management. The tables to be properly functional need to comply with the third normal form. To do this we used high level approach to the matter of the subjects, to whose we applied the following:

* Think of the entities as a self containing units
* Discover attributes based on the requirements
* Apply known methods to improve functionality
* Follow table design rules up u to the 3NF

As a result we got set of tables which were starting point for creating PHP classes. During the revision the tables design was modified according to our point of view and assumptions regarding to: ease of implementation of application code, satisfy actual relations between entities and also efficiency of working solution.

**Normal forms for tables design**

1NF:

* contains only atomic values
* there are no repeating groups
* primary key is set.

2NF:

* is in first normal form
* all non-key attributes are fully dependant on the primary key (no partial dependencies).

3NF:

* is in second normal form
* all non-primary fields are dependant on the primary key (no transitive dependencies).

Presented below images show the initial appearance of the tables, their transformation and final structure.

**Members table.**

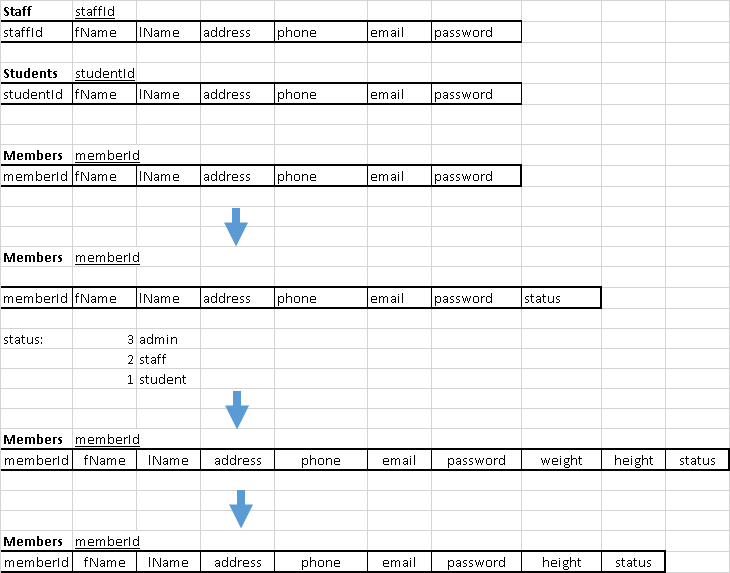
This table is going to store the details of the persons that will be allowed to use the application. Initially we distinguished three sets of users whose details will be stored (later admin, trainer and member). As we spot that the attributes of each set are shared and only level of privileges will differentiate between them, we decided to merge the tables into one and add one column to it called ‘status’. We also agreed on the meaning of the values stored by this field which are:

1 - student

2 - trainer

3 - admin.

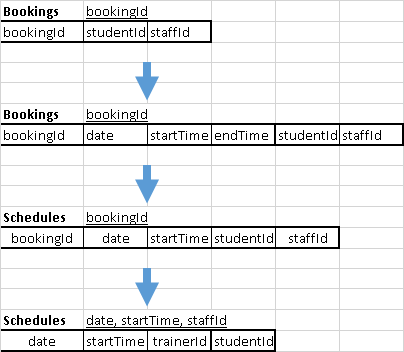
Last changes introduce two additional fields: height and weight which are used for calculating users’ BMI index. As we need to track the ‘weight’ over the time we decided to move it to the BMI table.



**Schedules table.**

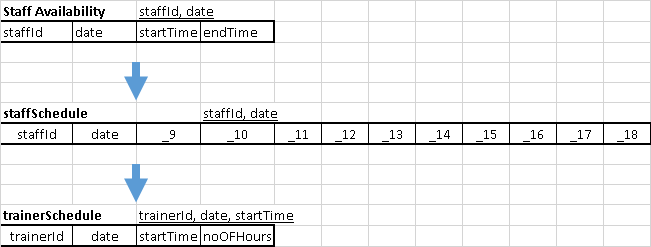
This table started as a link for the student and staff tables. Because the previous joined into single table it became a connection between rows of the Members table. The studentId and staffId (later trainerId) were supplemented with the date, startTime and endTime of the booking. Further revision discovered redundancy of the endTime column as for the most cases booking time for training sessions is one hour. Final inspection for this table removes bookingId field

which served as the primary key. This applies to other tables, as we will further see there is no need for additional field (bookingId) when set of existing fields can act as composite primary key.



**trainerSchedule table.**

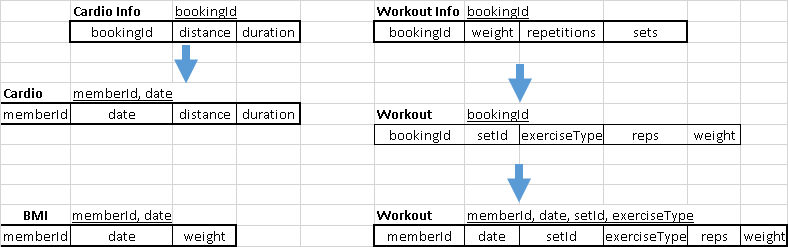
The design for this table represents quite significant changes over the time. From initial solution we moved to horizontal decomposition of hours which aimed for ease of write, read and manipulation of stored values. It became quickly known this is bad way to go because of unnecessary waste of space for the time when trainer is not working and also in the case of future changes i.e. extending of the opening time, the table would have to grow horizontally. This would violate good design principles.



**Cardio, WorkoutInfo, BMI tables**

These tables are additional artefacts that allow for enhancement of user experience.

* Cardio table stores details of distance and the time it was achieved.
* BMI is a simple table holding values needed for the BMI index calculation.
* The workout represents results of weight training.

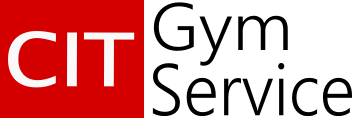
  
  
The consistent summary of the database tables design is presented in the following table:

# 

# Modeling and prototyping

## User Interface

## Logo Ideas



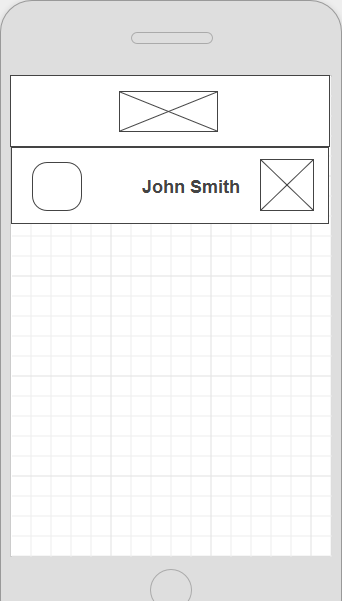


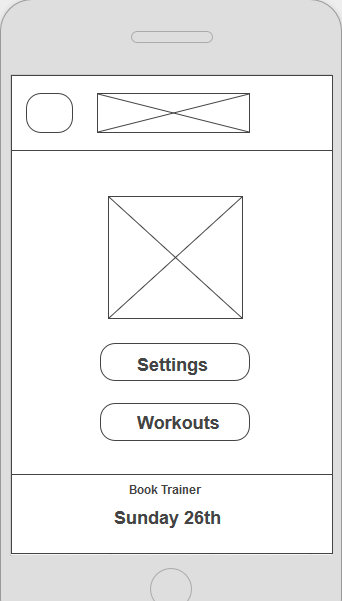
Ultimately we stuck with the standard CIT logo that’s used in other CIT websites.

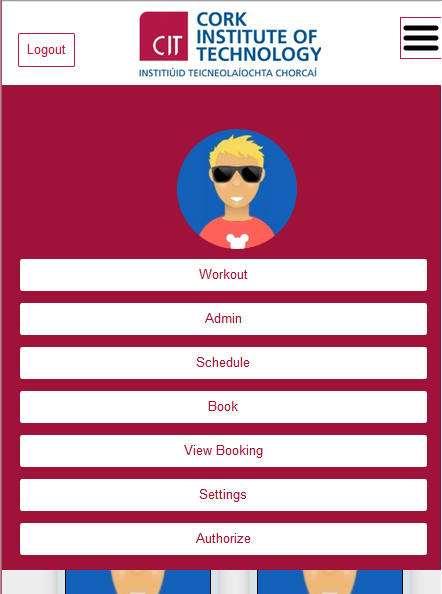
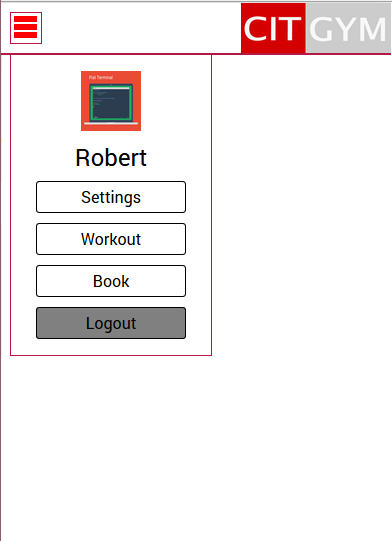
  
  
  
  
  
Old weebly prototypes here

Here content about html from John T’Player

## Mobile design

Evolution of the responsive design:





# 

# Code implementation

## HTML and CSS

?

John and Micheal Here

## PHP and SQL

Coding MVC



# 

The Model-View-Control (MVC) pattern, originally formulated in the late 1970s, is a software architecture pattern built on the basis of keeping the presentation of data separate from the methods that interact with the data. In theory, a well-developed MVC system should allow a front-end developer and a back-end developer to work on the same system without interfering, sharing, or editing files either party is working on.

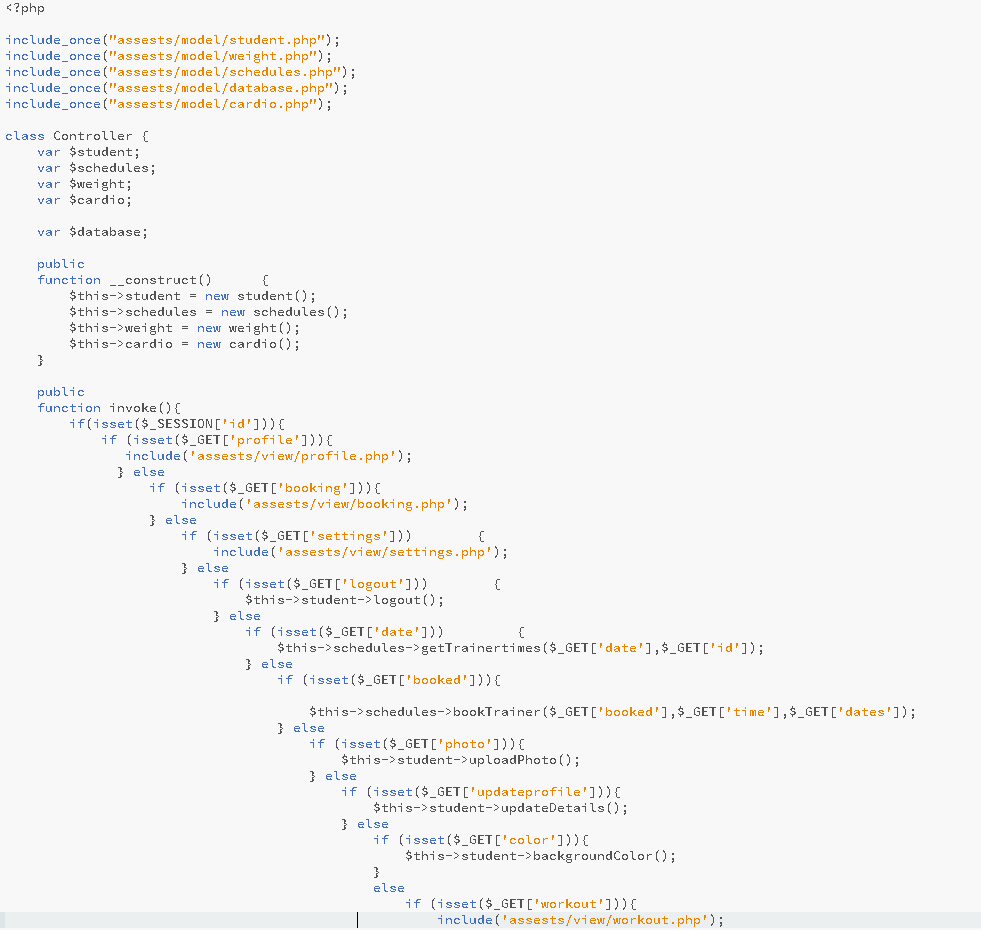
So we decided to try and build our own mvc framework. So that we can allow for different people to work on different files.

As seen on the diagram above, it allows us to work on different parts of the website alongside each other. We hosted the website on Github to allow for comments and bugs alerts.

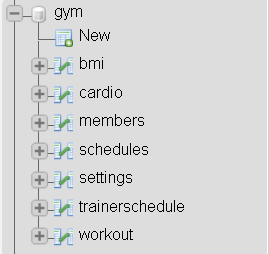
Created folder structure is as follows:



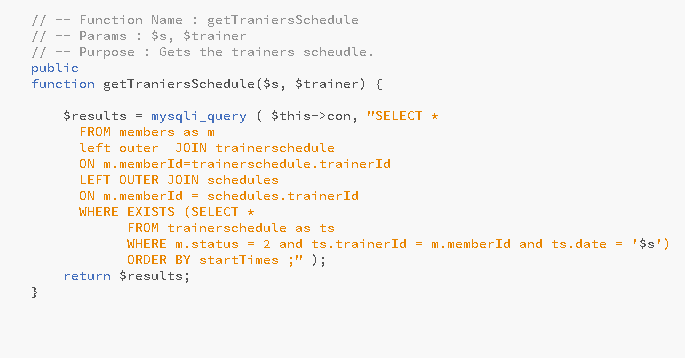
The example below represents the snippet of code from the controller class. It shows the structure of how the application responds to user’s actions in terms of navigation. It checks the the value of session variable to recognise that user is logged. It also redirect content to be displayed depending on values passed to attributes in the browser’s address bar.



For the application to be able to store information we had to support it with database. We decided to use MySQL that is shipped in package together with apache HTTP server. This was a convenience for us because we were developing our application on local machines and also this is the package that we are the most familiar with. We developed a database called ‘gym’ that contained set of tables presented below:



Reacting to user input involved use of using wide range of SQL statements: INSERT, UPDATE, DELETE and SELECT. For the most of the time we were operating with simple queries where given values were passed into database to be stored/updated or used for deleting/retrieving some information. There were few SQL queries with more complicated structure. Example of it is shown on the picture below. It represents three table join statement that pulls information from members, trainerSchedule and schedules tables based on provided values for the date and trainer Id fields.



# External code implementation

## What is an API?

An Api allows a software application to communicate with a remote application over the Internet through a series of calls and responses.

With APIs, the calls back and forth between applications are managed through something called Web services. Web services are a collection of technological standards and protocols, including XML or in Json , the programming language by which applications communicate over the Internet.

### Google Calendar API

### The Google Calendar API allows a program to perform many of the operations available via Google Calendar web interface. Using this API, it is possible to search for and view public calendar events. Authenticated sessions can access private calendars, as well as create, edit, and delete both events and the calendars that contain them.

Sites or applications that wish to have a deeper integration with Google Calendar can leverage the Google Calendar API. Such an integration could be a web application that creates or displays Calendar data. It could be a desktop application that synchronizes a user's Calendar with an existing desktop application. It could even be a device that brings the Calendar experience to a new platform.

https://developers.google.com/google-apps/calendar/concepts

**How did we use it ?**

We used it when a user books a training session of a trainer. It asks for permission to access their information and adds the event to the Calendar . The user will be able to get alerts and reminders to their phone of upcoming training sessions. The slight problem we ran into was that the system had to be authorized first before anything

### What are the Google Maps APIs?

The Google Maps APIs give developers several ways of embedding Google Maps into web pages, and allows for either simple use or extensive customization. There are now several API offerings.Create rich applications and stunning visualisations of your data, leveraging the comprehensiveness, accuracy, and usability of Google Maps and a modern web platform that scales as you grow.

In only a few lines of JavaScript code, build and style a map to call your own. With plenty of Google libraries and services at your disposal (including Geocoding, Directions, Street View and more) your imagination is truly the limit.

**How did we use it ?**

Well for a visual more inviting interface , we used the google map api to pick up the users location on the load of the home page. This api works if the user gives us the permission to access their location.

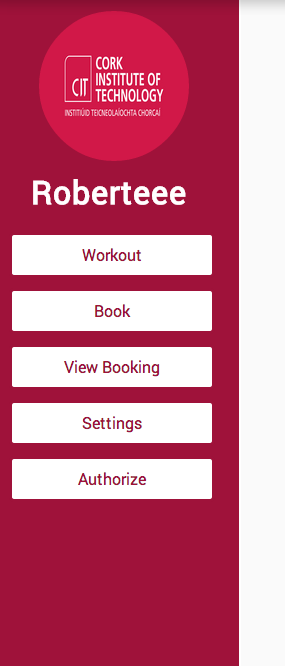
Highcharts



**What is it ?**

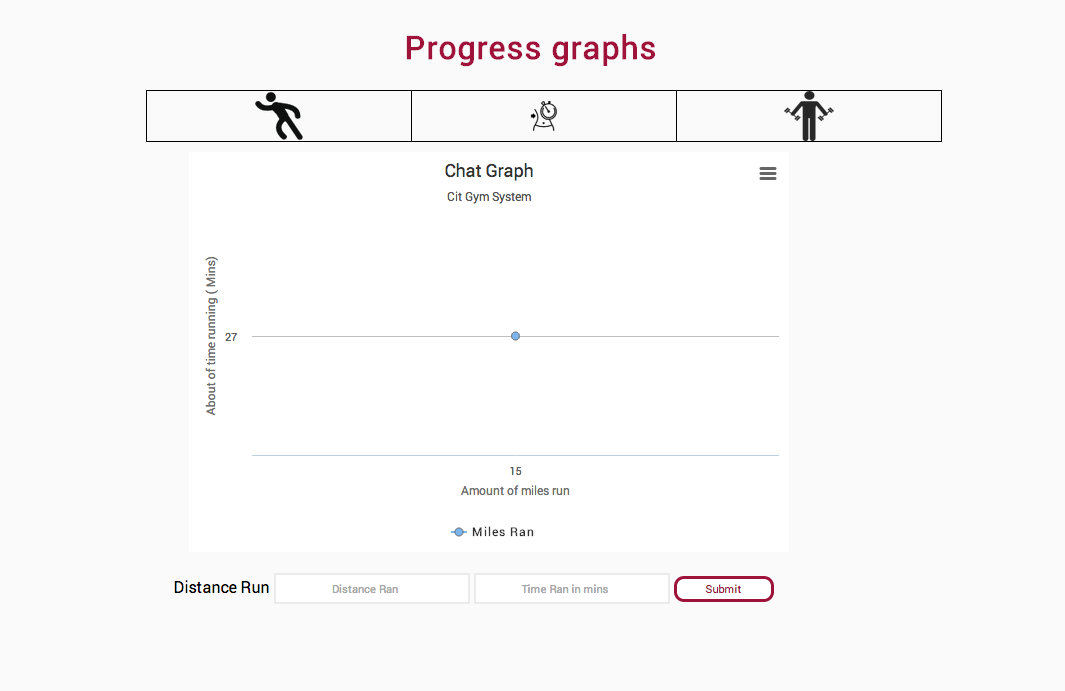
**Highsoft AS** is the company behind Highcharts JS, the JavaScript charting engine, and Highslide JS, the image and media viewer. Highcharts has quickly become one of the industry leaders in standards compatible, JavaScript based charting.

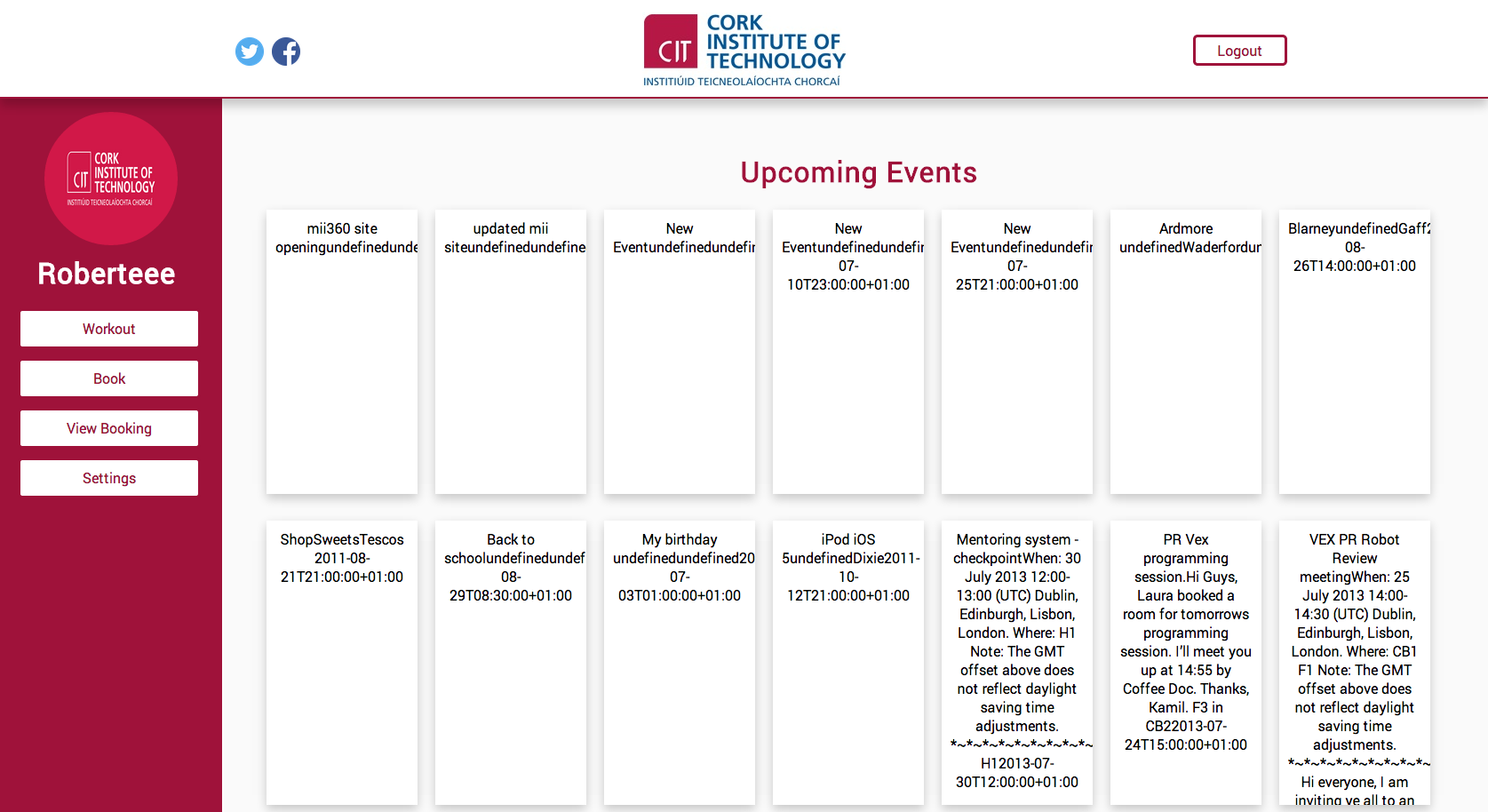
We use it in the displaying of information for the graphing system so users can track process in a great style and use. This can be seen in the workout section of the website.

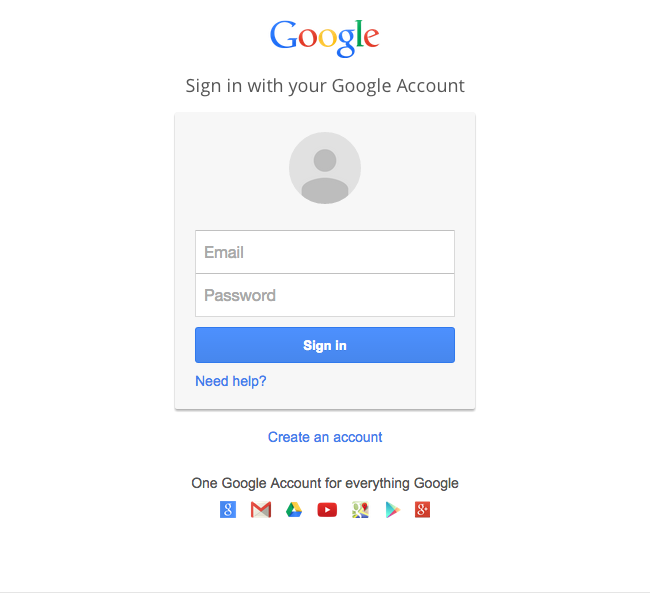


As you can see once a user has signed up and logged in they are welcome to the side menu, At the moment they have a authorize button, which allows you to sign in and link your google accounts to our gym account. A pop up window requests our system to allow information and access to your calendar and details. Such as user details, calendar events,birthdays and to allow to edit and update information in the calendar.

The use of the Google calendar api allows for greater room for future development to integrate the system into the current CIT system by the use of the Google account.







# Code testing

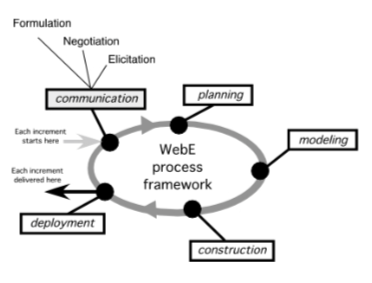
**For code testing , we used the agile form of testing which is when you test the software as you go. As we built the software , we tested . Making sure performance and sql was correct.**

**We ran use causes as we built to make sure the software was correct getting it done as we needed too .**

**The benefits of the agile testing in our cause it aloud for better communications between all of us Because the tester and developers are working on the same team, they can more easily communicate with each other.**

**Along with we were able to give each other immediate feedback which lets us fix the bug while the code is still fresh in our mind.**

**As you can see below we used the Web E process framework , to build and deal with problems as they arise.**



Part 3 - **Project Management and Communication**

# Initial assessment

# Work organization

# Project progress

For the purpose of this project we decided to schedule our team meetings on weekly basis with Thursday as a day that fits into our supervisor and our time schedule. Regular meetings helped us to ease communication within a team, to plan and set goals for next task increment and to track overall progress. With this approach we were able to review our work, discover issues and apply solution early, before we step into the next task. Team meetings gave us as a group and also as individuals the possibility to establish common spirit and deepen the attitude geared toward collaboration and help to other members of the group. At each meeting we:

· discussed the details of previous task up to the point

· reported problems that need to be addressed

· collaborated on the elements that need common agreement

· planned deliverables for the next week

· asked our supervisor for clarification in case of any confusion

**Meeting 1 – 2nd Oct 2014**

Today, after organising our team, we met up and discussed the specification for the project. We came up with our own ideas for the project. After agreeing upon what idea suits the best we then spoke about the high level details of proposed web site design and decided who will work on particular aspects of the developed application.

The plan for the next week:

· Analyse business needs and context of work that has to be done.

· Define knowledge and skills necessary for solution development

· Set initial environment and ways of communication

**Meeting 2 – 8th Oct 2014**

Within the last week we have established the list of events and the constraints that are related to the gym business and developed application. As this is the early stage of project we were mostly working together. The results of our work are sufficient, in our opinion, to proceed into modelling and prototyping part. While scheduling tasks for next week we are going to leave some margin for unknown as more than often details of unexpected could appear.

Tasks for next week:

· Finalise the requirements

· Define high level wireframes that represent user interface details

· Create HTML structure to support design

· Model database tables an start work on PHP classes

**Meeting 3 – 16th Oct 2014**

On today’s meeting we have discussed issues with our tables design. Apparently the most problematic one was the **trainerSchedule** table. We had three different concepts on how to store information about scheduled working time for the trainers. After consultation with our supervisor we made the decision to drop third idea and pick one of the remaining two that seems to be the optimal solution. We also talked about the representation of the days of the week by using calendar form. This will be used with external library that allows for reading the values when user clicks on particular day field. At this point we were also happy with the wire frames created and the HTML code that represented it.

Tasks scheduled for next week:

· Implement PHP code for classes and database connectivity

· Integrate calendar feature

· Continue work on the user Interface.

**Meeting 4 – 23rd Oct 2014**

Today we have reviewed working prototype of user login and profile page, and proposed only small changes to the layout and colour scheme. The database design was also finalised. The implementation of server side code delayed our work because of problems with sharing parts of the code that interfered with each other. To solve it we decided to separate the work over the code between application and database.

Tasks for next week:

· Connect server side functionality with front end design for login, signup and profile page

· Work on CSS styling in regard of responsiveness and user usability

· Implement ‘book’ and ‘schedule’ pages and google calendar API

**Meeting 5 – 30th Oct 2014**

Within last week we identified major issue with the code sharing and tracking its multiple versions. Managing rapid changes brought a bit of confusion, so as a solution we decided to place our project on the online source core management (github) and store our code in its repository. We are aware that this imposes need for extra time and effort to be spent on familiarising with the new application but, at the end it will allow for our project work to be completed. The next meeting we are going to schedule in two weeks’ time.

**Meeting 6 – 13th Nov 2014**

Today we discussed the issues for mobile version of our design. The look of the website did not work well on small device resolution the decision we made was to give it more effort and if still in troubles seek for help of bootstrap framework. Also we encountered a problem with the displaying of trainers’ schedules in the ‘book’ page. The results of database querying caused the details of each trainer to be mixed with other. We will have to fix the query structure or simply it and add extra code in order to manage returned results. The one more problem we challenged with was linked to trainers scheduling functionality. We will need to look closer to possible solutions for this.

Tasks for next meeting:

· Fix SQL query

· Improve responsiveness

· Implement admin area

· Decide on trainers’ flexible hours and apply solution

**Meeting 7 - 27th Nov 2014**

For the last two weeks we have solved the SQL problem. We were able to tweak it so the returned results could be properly displayed. In mean time we also worked on the details of the user interface. As it turns out front design is a live element of our project and we have to review the impact of any changes we want to introduce before they can be applied

# Project issues and solutions

Validation of user input

responsive design

sql

meeting with the guy

code sharing

# Learning outcome and final conclusions

# 

# Appendix:

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 