

**Gym Management and Administration System**

This Report is submitted in partial fulfilment of the requirements for the

BSc (Honours) Information Systems and Information Technology (DT249) to the

School of Computing,

College of Sciences & Health, Dublin Institute of Technology.

**Author:** Tomas Mikoliunas – D14123810

**Supervisor:** Cathy Ennis, School of Computing

**Date:** 27 December, 2016

**Declaration**

I, **Tomas Mikoliunas** hereby confirm that the project I now submit for Honours Project Module, titled **Gym Management and Administration System** is my own research work and was written by me following my research. The work is new and has not been submitted for any previous award. I further confirm that the work has not been taken from the work of others save and to the extent that such work has been identified, cited and acknowledged within the text of my proposal and presentation.

**Signed:**

**Date:**

# Acknowledgement

I would like to thank my supervisor, Cathy Ennis, for her help and guidance provided during the course of the project, who provided much needed support, encouragement and understanding through the long days require to complete my project.

# Acronyms, Abbreviations, and Definitions

|  |  |
| --- | --- |
| JDBC | Java Database Connectivity |
| JSP | Java Server Pages |
| MVC | Model View Controller |
| JSON | JavaScript Object Notation |
| POJO | Plain Old Java Object |

Table 1. Acronyms, Abbreviations.

|  |  |
| --- | --- |
| Administrator | Someone who works in the gym and who has special permissions in order to control and manage the system. |
| Member | The registered member who uses the facilities of the gym club and also can interact via personal account provided by gym’s website. |
| Guest | A person who just can visit the gym’s website, check available prices and discounts, etc. and who is not entitled to login. |

Table 2. Definitions

Table of Contents

[Acknowledgement 2](#_Toc469315764)

[Abstract 4](#_Toc469315765)

[1. Introduction 5](#_Toc469315766)

[1.1. Background 6](#_Toc469315767)

[1.2. Project Objectives 6](#_Toc469315768)

[1.2.1. Limitation of existing systems 7](#_Toc469315769)

[1.3. Project Structure 7](#_Toc469315770)

[2. Technology choices and evaluation 8](#_Toc469315771)

[2.1. HTML for the web application 8](#_Toc469315772)

[2.1. CSS for defining the layout of HTML documents 9](#_Toc469315773)

[2.2. JavaScript, JQUERY for the dynamic web development 10](#_Toc469315774)

[2.3. AJAX for the asynchronous and synchronous calls without reloading the web application 10](#_Toc469315775)

[2.4. Spring MVC framework for the model-view-controller architecture development and dependency injection 11](#_Toc469315776)

[2.5. MySQL Relational Database Management System (RDBMS) 13](#_Toc469315777)

[2.6. JDBC Template, a database connection mechanism and for SQL queries execution 14](#_Toc469315778)

[2.7. Spring Security – Password Hashing or Password Encoding 14](#_Toc469315779)

[2.8. Apache Tomcat web application server 15](#_Toc469315780)

[3. Application implementation and design 16](#_Toc469315781)

[4. Testing 16](#_Toc469315782)

[4.1. Unit testing 17](#_Toc469315783)

[4.2. Unit integration testing 17](#_Toc469315784)

[4.3. Validation testing 18](#_Toc469315785)

[4.4. Performance testing 19](#_Toc469315786)

[4.5. User testing 19](#_Toc469315787)

[5. Conclusion 20](#_Toc469315788)

[6. References 20](#_Toc469315789)

# Table of Tables

[Table 1. Sequence number to the requirements 22](#_Toc469683328)

# Table of Figures

[Figure 1. Login page 11](#_Toc469676515)

[Figure 2. CSS style 12](#_Toc469676516)

[Figure 3. Drop-down menu implemented with JQUERY 13](#_Toc469676517)

[Figure 4. Submit the form 14](#_Toc469676518)

[Figure 5. Workflow of the Spring web MVC. 15](#_Toc469676519)

[Figure 6. Tables in 'gym' database 16](#_Toc469676520)

[Figure 7. Data Source for MySql connection 17](#_Toc469676521)

[Figure 8. Insert new Admin user into database. 17](#_Toc469676522)

[Figure 9. 'encode()' function for password encryption 18](#_Toc469676523)

[Figure 10. encrypted password 18](#_Toc469676524)

[Figure 11. Configure Java Buid path in STS. 19](#_Toc469676525)

[Figure 12. Configuring Tomcat for STS. 20](#_Toc469676526)

# Abstract

This Project will look at Gym Management System which will be easy to use gym club management system. It is my intention to develop a secure which will consist of two modules: Main Web Application and a Database. Objective of my project is to provide an easy to use Gym Management System package. In terms of success criteria what would it contribute in business, it is my intension to provide the system. The web application will be password protected, so only authorized person could use this system.

The report comprises of an introduction which describes the project aim, objectives and scope followed by a technology overview, requirements detailing and finally the technical and design implementation.

# Introduction

The Health and Fitness Industry is becoming a more and more competitive market [1]. This makes knowing your key point of difference ad communicating it to your target market, vital for survival. At its essence your point of difference defines not only what your offering is, but who you offer it to and how you present it.

As a gym owner, health club operator, or as a freelance personal trainer, without a doubt, one or more following questions crosses the mind daily…:

* How much membership plans did our gym sell yesterday/last week, etc.?
* Which one of our membership plans or products is selling the most?
* What is my gym retention rate?
* Who cancelled today and what was the total losses?
* Etc…

An age old business adage states *“what gets measured gets improved”*. Therefore, we should measure the things that are important to your business [2].

As a gym owner who is running the business should be aware of when he/she was in danger of posting a bad week, month, or quarter way before the numbers were crunched by accountants. Having these statistics in advance is very good for a business. .

This Project will look at Gym management system which is an easy-to-use gym and health club membership management system. This software will help the gym administrator to keep records of members and their memberships, and have quick and easy communication between administrator and gym members. This Gym Management Software also includes user-friendly administrative interface, booking system, point-of-sale, accounting, which will help in management of the club, for example, payment and due collection, create, view, edit, delete member details or membership plans, member plan expiration and accounts status, photo capture supported, members schedule history and a range of reports that help in the management of the club.

I am attracted to this project for a number of reasons. The best gym management software should allow you to create, edit and easily find a member in the system. In the end, it should convert all tedious manual organising into automated, responsive cloud-based software. In addition to all the basic membership management features, my Club Management Software has dashboard that help to see how the organization operates over time, provides online sign ups, online booking system, inventory and point-of-sale system. Software’s strong focus on events and also the ability to see the success the organization long term is great for the business.

This is when I came up with an idea. This is still very new to me and my goal is to leverage from this college project in order to become a good analytic and developer.

## Background

My Gym Management Software is a complete gym and recreation facility system program which looks after all the members, memberships and activities. It is designed for gyms and health clubs.

My Gym Management Software provides lots of functions such data entry of club members, keeping records of all the things about club member’s fees, plan, bought or reserved products, physical fitness which help to provide good quality of services to club members from Gym administrators. Data to be stored in the database.

This system structure is become very simple to understand because of Data Flow Diagram provided by me. Context level Diagram and some chart are also available in the case study.

## Project Objectives

The Objective of my project is to provide the Software for helping gym owners, health club operators, freelance personal trainers, etc. better grow their businesses and promote their causes. Its goal is to make a complete membership solution that contains many enterprise features.

In terms of success criteria what would it contribute in business, it is my intension to provide system that would simplify the business of the club:

* Manage and maintain the member history easily
* Manage memberships management and packages
* Easy-to-use software which handles the customer-staff relationship in an effective manner
* Also, only the authenticated users have the privileges to access any database and make the required changes, if necessary
* Provide fast check-in process for members
* Online sign ups
* Online booking system
* Inventory and point-of-sale system
* E-mail campaigns
* Help to reduce the overload paperwork
* Minimize the human mistake and error, due to computerization, to provide validation for all operations
* To develop a user friendly system that requires minimal user training

### Limitation of existing systems

Being ‘customer-focused’ is a claim that many businesses make, but few actually deliver on. Not to say that most businesses have poor customer service, although this can be the case, but that every business provides some level of customer service. To qualify as a point of difference, means not simply solving the issues as they arise, but pre-empting them and being actively involved with clients. Executed correctly, it can create an incredibly loyal customer base.

Recall from one gym club owner: “*I have two gyms right now operating with a membership management software called Gym-master.. But I can only use it for a database reference for member retention rather than a real-time visualizer for all aspects of my business*” [2].

Also feedbacks followed with:

* The existing System such as Gym-master is not as much as user (Customer) friendly
* The communication with members is not well in Existing System because all the data is handled by Gym manager
* Members don’t get full accessibility to Gym centre and all permissions are allowed only for Gym manager
* The system cannot take effort out of finances and debt collection
* Systems are time taking software and cannot be easily install in Operation Systems

## Project Scope

I am going to describe the division of entire architecture in to two modules – Admin (administration) and Customer (Member of the Gym club).

The operations that can be executed by an admin are search records, modify records, delete records, add records, generate dues and various statistical reports and view. All these will be shown will be shown through a diagrams provided at following chapters of the report.

The operations that can be executed by the customer is viewing the website of Gym club, membership packages and prices, viewing all promotions and discount, viewing of personal profile, updating personal details, viewing personal account history.

## Project Structure

This report is divided into a number of chapters, each chapter will detail a specific area of the project.

The early sections detail the technology that was used and why it was used, followed by a specific details involved in the application design and construction with many relevant code examples. Finally, the report will show my conclusion and what I felt, I have learned over the course of this project.

# Literature Review

## Introduction

This chapter focuses on the literature overview and research carried out in the field of the project to gain a better understanding of the architecture used to build my web application and the components of this architecture that will be used for the project.

For me to create an application for a Gym Management System which would be deployed on a web server and would be displayed on the web browser, a number of technologies were required, each involving their own specific learning challenges. While we did cover some of these technologies over the course of this program, the outcome was more foundational level knowledge. After day to day use and practice I would now present the following technologies were utilized to develop my intended project:

* Unified Modelling Language
* HTML used as the format of the web application
* CSS the style language that defines layout of HTML documents
* JavaScript, JQUERY used in web development to manipulate the web page elements and to make it dynamic
* AJAX which is also a technique for creating fast and dynamic web pages, and which allows web pages to updated asynchronously by exchanging small amounts of data with the server behind the scenes, without reloading the web page
* Spring MVC framework which provides model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications
* Spring Security
* JDBC Template, which is a powerful mechanism to connect to the database and execute SQL queries
* MySQL, a database technology, Relational Database Management System (RDBMS)
* Apache Tomcat web server and servlet container that is used to serve Java applications
* RESTful Web Services

## Unified Modelling Language

The Unified Modelling Language (UML) is a set of modelling conventions that is used to specify or describe a software system in terms of objects. An industry standards body, the Object Management Group (OMG) adopted the UML in 1997 [3]. It provides a graphical syntax for an entire series of object models. The different diagrams each show a different perspective of the information system. The various modelling techniques of UML comprise, Use Cases, Class Diagrams, Sequence Diagrams, Package Diagrams, State Diagrams, Activity Diagrams and Deployment Diagrams. UML diagrams can be used in conjunction with many different processes or can be used as a standalone in the absence of a formal process [4].

I have chosen Use Cases, Class Diagrams and Sequence Diagrams to model my project. The diagrams will be shown and described in 3.2Use Cases.

The flowcharts will also be included in design for the graphical representation of some processes. The purpose of flowchart is to show how process is performed from start to finish, usually in sequential step order. Elements that may be included are: sequence of actions, materials or services entering or leaving the process (inputs and outputs), decisions that must be made, people who become involved, time involved at each step and/or process measurements. The Free Flowchart Maker – Lucidchart [5] software I have used to develop understanding of how a process is done.

## 2.1. HTML for the web application

For the supporting Gym Management System web application, it was required that I further increase my knowledge of HTML coding, reading book “HTML and CSS: Design and Build Websites” [6], with an endless supply of resources available online: w3shcools.com, tutorialspoint.com, etc.

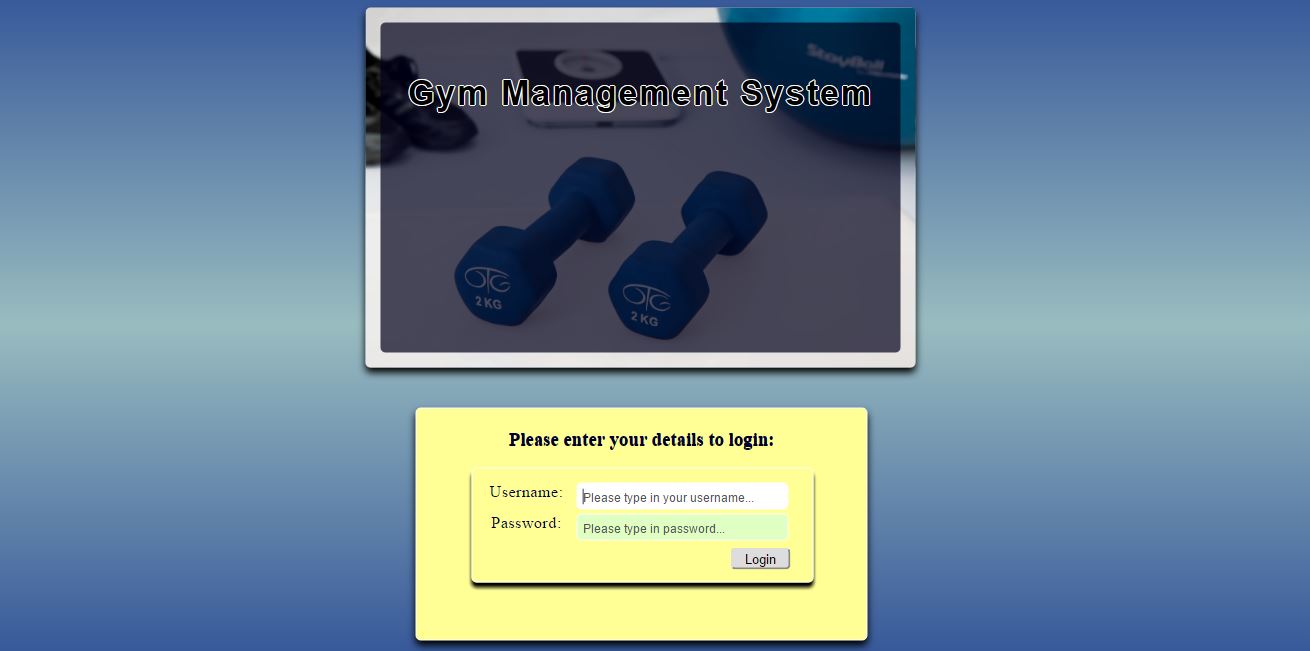


Figure 1. Login page

## CSS for defining the layout of HTML documents

I found that I was quickly able to build framework of my site, the more polished elements were achieved using cascading style sheets. Cascading Style Sheets or CSS allows for the styling aspects of a web-page to be controlled from a central file. By referencing a style sheet many web-pages can be styled from a central source. For my site I have written many files for typography, forms, buttons, navigation and other interface components (see Figure 2):



Figure 2. CSS style

## JavaScript, JQUERY for the dynamic web development

JavaScript and JQUERY were used to control web application and to run on the client side by the web browser. They support object-oriented programming and procedural programming. Thanks to my source books *“JavaScript step by step”* [7] and *“JavaScript and JQUERY: the missing manual”* [8]*,* where I have significantly deepened my knowledge about this programming language and its library. The first requirement for the web app was to define the content of the web page. This was done using HTML and CSS. Using just two languages like HTML and CSS you can create static web pages.

Incorporating JavaScript and JQUERY into a web application allows you to improve the user’s experience of the web application by converting it from a static page into one that can interact with them. With the usage of JQUERY, which is the library based on JavaScript, I have added animation into the web app which either attract attention to a specific part of the page or which make the page easier to use. I have provided responses within the web application to various actions, like drop-down menus, sliding windows, etc. (see Figure 3):

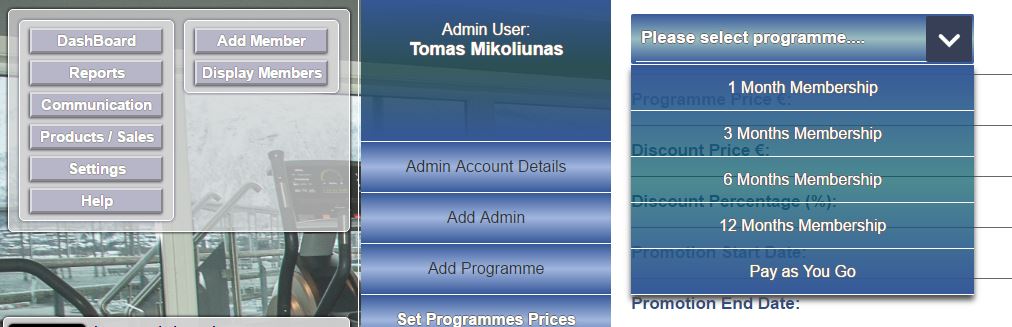


Figure 3. Drop-down menu implemented with JQUERY

## AJAX for the asynchronous and synchronous calls without reloading the web application

Using just two languages like HTML and CSS you can create static web pages that will be accessible regardless of which device is used to access the page. These static pages can interact with your visitor through the use of forms. Once a form is filled out and submitted a request is sent back to the server where a new static web page is constructed and eventually downloaded into the web browser.

The big disadvantage of web pages like this is that the only way that your visitor has of interacting with the page is by filling out the form and waiting for a new page to load.

After reading the book “Foundations of Ajax” [9] and doing exercises my self, I found that one of the great advantages, the AJAX technology does that by converting your static page into one that can interact with your application users without them needing to wait for a new page to load every time they make a request. Basically, AJAX is about updating parts of a web page, without reloading the whole page. I found it very useful for my one-page web application, where I have loaded different parts of the main page using AJAX calls to the server, which allowed me to update that asynchronously by exchanging data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

The example shows the ‘form’ submit with AJAX call to the server:

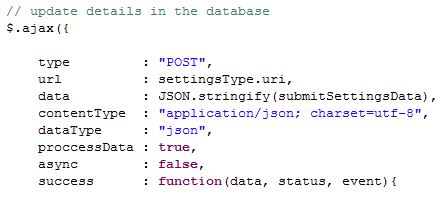


Figure 4. Submit the form

This technology was new to me, and I feel strongly enhanced my knowledge for the web development capabilities.

## Spring MVC framework for the model-view-controller architecture development and dependency injection

The framework like Spring MVC helped me to understand the way of building in Java the 3-tier web application. The MVC pattern results in separating the different aspects of the application (input logic, business logic and UI logic). MVC consists of Model, View and Controller. Model will be not only the data, but also the methods that will change this data.

I was new to Spring framework and since I had relevant knowledge of Java programming language and after my research on how your Java applications can benefit from this great framework and how it is in demand among the companies, I decided to learn this framework as it would help me to develop a great web application and also would be the advantage in my future job search.

During the research lots of readers suggested me *“Spring in Action”* [10]*,* as one of the best books to learn Spring. Seems like a worth reading book. I was looking for a common book to learn the both Spring MVC for web development and Spring core for using Spring framework as DI and IOC container, so *“Spring in Action”* is definitely a good choice.

Following is the list of few great benefits of using Spring framework:

* Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat [11]
* Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using JavaBean-style POJOs, it becomes easier to use dependency injection for injecting test data
* Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC into consistent, unchecked exceptions.

.Spring aims to make enterprise Java development easier and to promote loosely coupled code. Vital to this is dependency injection (DI) and aspect-oriented programming (AOP). DI is a way of associating application objects such that the objects don’t need to know where their dependencies come from or how they’re implemented. Rather than acquiring dependencies on their own, dependent objects are given the objects that they depend on. Because dependent objects often only know about their injected objects through interfaces, coupling is kept low. AOP enables you to centralize logic that would normally be scattered throughout an application in one place – an aspect. When Spring wires your beans together, these aspects can be woven in at runtime, effectively giving the beans new behaviour.

The Spring web MVC framework is designed around a DispatcherServlet that handles all the HTTP requests and responses. The request processing workflow of the Spring web MVC DispatcherServlet (see Figure 5).

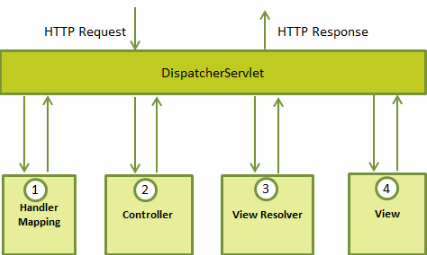


Figure 5. Workflow of the Spring web MVC.

## MySQL Relational Database Management System (RDBMS)

I chose MySQL as the database system as it is a popular, well documented and open-source system.

The MySQL database will store and manage all of member accounts, details etc.:



Figure 6. Tables in 'gym' database

“*In*[*computer science*](https://en.wikipedia.org/wiki/Computer_science)*, a database connection is the means by which a*[*database server*](https://en.wikipedia.org/wiki/Database_server)*and its*[*client*](https://en.wikipedia.org/wiki/Client_(computing))*software communicate with each other. The term is used whether or not the client and the server are on different machines*.” [12].

To establish and maintain communication with the database, the JDBC Template (Java Database Connectivity) of the Spring framework was used. This going to be discussed in the next section.

## JDBC Template, a database connection mechanism and for SQL queries execution

The JdbcTemplate class executes SQL queries, update statements and stored procedure calls, performs iteration over ResultSets and extraction of returned parameter values. It also catches JDBC exceptions. Instances of the JdbcTemplate class are thread-safe once configured. So you can configure a single instance of a JdbcTemplate and then safely inject this shared reference into multiple DAOs. A common practice when using the JdbcTemplate class is to configure a DataSource in your Spring configuration file, in my case **spring-servlet.xml** (see Figure 7) and then dependency-inject that shared DataSource bean into your DAO classes [13].

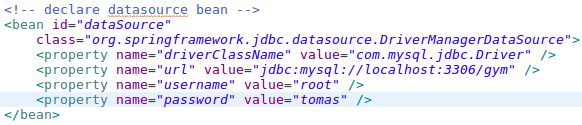


Figure 7. Data Source for MySql connection

We can perform all the database operation by the help of JDBC Template class such as insert, update, delete, and retrieval of the data from the database.

The example below (see Figure 8) shows simple SQL query to insert new Admin user into database:

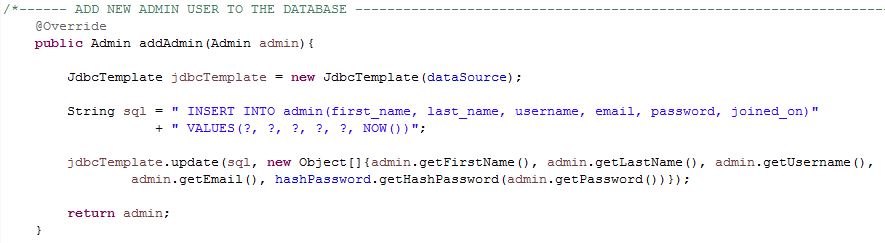


Figure 8. Insert new Admin user into database.

## Spring Security – Password Hashing or Password Encoding

Thanks to Spring Security [14] I used to intercept sensitive URL requests (like admin or the member will not be able to access application content that are only designed for using by authenticated users). Again, this feature was new to me and I feel significantly improved my development skills, knowing that security against unauthenticated users is vital of this type of web applications.

This is very useful feature of Spring where I was able to integrate Spring Security with a Spring MVC framework. This feature gave ability to encode the login passwords for the admin and customers and store it into the database.

Saving your password in a simple string form in database is never a good idea, instead one must save his password in some sort of encoded or hashed format. To make password purely protected we must have some algorithm to identify how to save encoded password in database and how to authenticate user using an encoded password stored in database.

To encode a password using 'BCryptPasswordEncoder'[14] spring-security provides ‘encode()’ function of 'BCryptPasswordEncoder'class (see Figure 9):



Figure 9. 'encode()' function for password encryption

Using the ‘BCryptPasswordEncoder’ and ‘encode()’ function, the password received has been encrypted.

And I test it that the password was encrypted with the ‘Logger’ [15] class provided by Spring (see Figure 10):

C:\Users\Tomas\Documents\Koledzas\Honours_Degree_Project\IPR-2_Assignment_(5%)\encrypted_password.JPG

Figure 10. encrypted password

## Spring Tool Suite (STS) for Java programming

Using Integrated Environment (IDE) was started during mid-1970s and has been gradually evolved to extremely comprehensive tools and environment since then [16]. This has enabled developers to perform their activities in a more efficient way. However, although programming without using IDEs can be a very cumbersome job, from one side, using IDEs needs familiarity with the tool, on the other side.

Overall, using IDE would give me almost all the tools that I would need to develop my system, such as project creators, smart editors, compilers, debuggers, testers, and more, in one package.

For my project development I have chosen to use the Spring Tool Suite (STS) [16] IDE to handle all levels of Java development. The Spring Tool Suite (STS) provides the best Eclipse-powered development environment for building Spring-powered enterprise applications. STS supplies tools for all of the latest enterprise Java and Spring, and comes on top of the latest Eclipse releases.

Before I started development I needed to make sure that the appropriate version of JDK is configured in my environment. In my STS environment just clicked on ‘Project Properties -> Java Build Path’ and added all necessary libraries and jar files (see Figure 11):

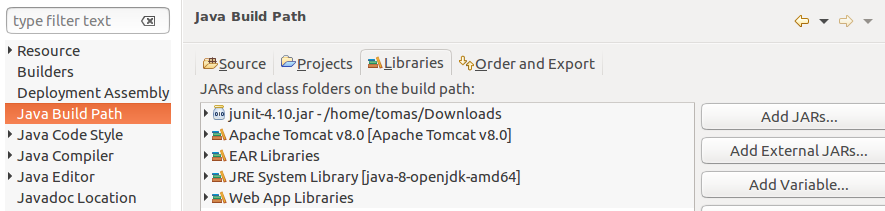


Figure 11. Configure Java Buid path in STS

## Apache Tomcat web application server

Apache Tomcat server [11] is a servlet container that is used to serve Java servlets, that is Java EE web applications run within Java EE application servers and web containers (also known as servlet containers).. Tomcat is a server that is meant to run applications that were written in Java. Every application server contains a web container, which is responsible for managing the life cycle of servlets, mapping request URLs to servlet code, accepting and responding to HTTP requests.

The servlets are Java classes that are used to serve dynamic web content. You can use Tomcat to serve static content as well. Recent versions of Tomcat have an improved performance to serve static content.

Tomcat is an open source implementation of the Java Servlet and Java Server Pages technologies, released by the Apache Software Foundation [17].

After successfully downloaded and installed the Tomcat server I needed to configure it to work it properly in my STS [16] environment, where I selected the Apache installation Directory in my STS and clicked ‘Finish’. Under ‘Servers’ tab now I could see ‘Tomcat v8.0 Server at localhost [Stopped, Republished]’ and double clicked to verify HTTP ports information. The default HTTP port is 8080, which I am going to use (see Figure 11):

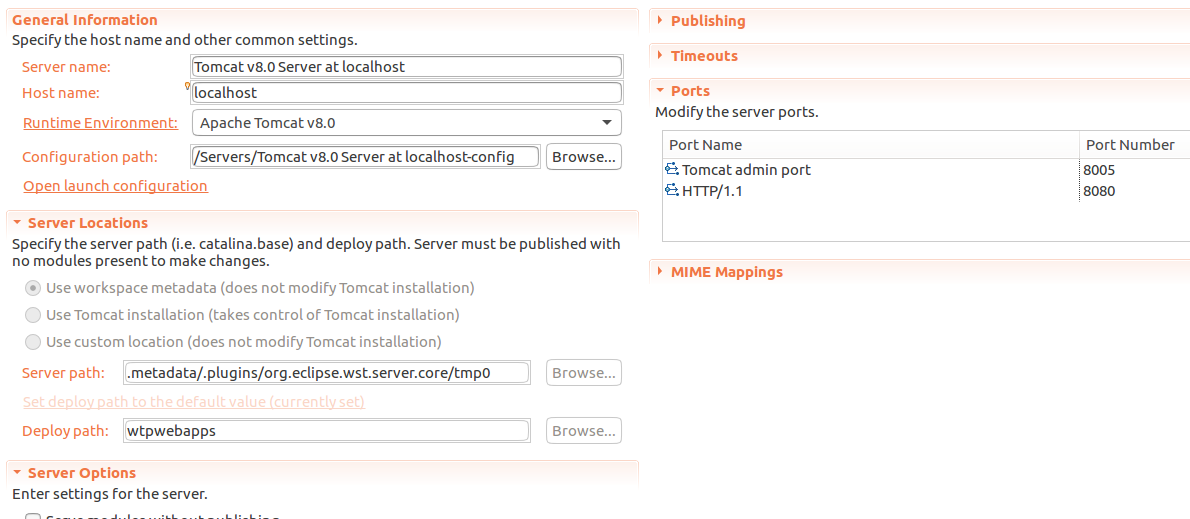


Figure 12. Configuring Tomcat for STS

To start the server just right-clicked under ‘Servers’ tab and ‘Start’.

## RESTful Web Services

In order to facilitate communications with business logic and the database a REST API [18] has been utilized. REST stands for Representational State Transfer. REST is an architectural design offered by service providing web sites. Web service is a way of communication that allows interoperability between different applications on different platforms and different programming languages. The communication can be done through a set of messages over HTTP protocol.

Web services are browsers and operating system independent service, which means it can run on any browser without the need of making any changes. Web Services take Web-applications to the next level.

In recent years RESTful web services have been growing in popularity, this is because tech companies are becoming aware that having a Serviced-Oriented Architecture (SOA) is essential in a competitive industry. REST works by allowing a method to be called via a HTTP request essentially exposing a software service to the internet.

Web Services allow the business logic of many different systems to be exposed over the Web. This gives your application the freedom to chose the Web Services that they need. Instead of re-inventing the wheel for each client, you need only include additional application-specific business logic on the client-side. A response is rendered and returned to the invoker (see Figure 13. REST diagram).

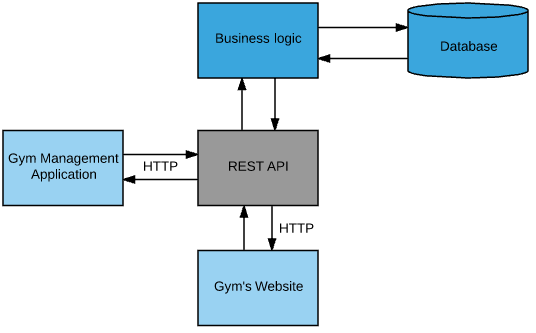


Figure 13. REST diagram.

# Design

## Introduction

Analysis is the process of understanding and defining what functions are required from the system and identifying the constraints of the system operation and development. It is considered to be the critical stage of the software process. Errors that occur at this stage inevitably lead to later problems in the system design and implementation.

Comprehensive analysis and design of the application is covered in this chapter, the technical implementation as well as the visual components.

In this chapter the specific user requirements of the project are presented. Use case diagrams are used to give a graphical overview of the uses. Each use case is presented with a description, the actor it is considered with and the flow with which the actor would interact with this use case.

## Analysis

Analysis is considered to be the most critical phase of a project [4]. It has been said that it is better to spend more time analysing and designing a solution than coding the implementation.

To start development of my project, initially for me was necessary to investigate what the application should do to meet user requirements, what are functional requirements, and I had to prioritize my requirements in order to make an agreement on the sequences of the implementation process, i.e. what are the core functionalities without which the application cannot be utilized? What is the sequence of dependency between different features? These questions and their answers helped me to plan for the iterative and incremental development, as this was highly crucial for me to know what I am going to do.

### User Interface

* HTML has been used for developing the User Layout for the system.
* JavaScript and JQUERY has been used for creating all the validations and client side scripting functionality.
* CSS and Bootstrap has been used for designing the web pages of the system.

### Software Interface

* **Client on Internet**: Web Browser, Operating System (any).
* **Web Server**: Apache Tomcat, Operating System (any).
* **Database**: MySQL.
* Languages: Java, HTML, JavaScript, JQUERY
* Framework and Technology: Spring/Spring MVC, AJAX

### Communication Protocol

Following protocol is required to be permitted on the server side:

* HTTP incoming requests.

### Functional Requirements

I have prioritized the functional requirements based on two factors, their importance and their sequence.

For the importance, I have used the following labels:

1. **Must have** – the requirements without which my application would not provide the minimum capability that the user expects.
2. **Should have** – the requirements that are not part of the core functionalities of my application, however, they can be very useful and I would try to implement them if the project schedule allows me to do so.
3. **Nice to have** – the requirements that they do not play a main role in the functionality of the system, at least at the scope of this project, however, they would give more facilities to the users if the schedule of the project allows them to be implemented.

For the sequence factor what matters to me was to choose a sequence that would show the route of my development (see Table 1. Sequence number to the requirements):

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Importance |
| 1 | The administrator of the gym club must to login to start the session, by providing the username and password. | Must |
| 2 | The member of the gym club must to login to the gym’s website, in order to perform personal account actions, by providing the username and password. | Must |
| 2 | The password recovery must be implemented and allow users to change their password. | Must |
| 3 | The administrator entitled to perform all the actions included in the system software package. | Must |
| 4 | Create, retrieve, modify, delete the information in the database | Must |
| 5 | Display pricing and stock information, filter by product type | Should |
| 6 | Place orders, calculate sales total, create a sales record | Should |
| 2 | To keep and maintain gym members information. The members should be classified | Must |
| 3 | To provide repots on statistics, payment due | Must |
| 4 | To maintain and handle reservations | Should |
| 5 | To notify members on promotions via emails | Should |
| 6 | To notify members on availability | Should |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table 3. Sequence number to the requirements

### Non Functional Requirements

* Performance: System (including the web site) should be able to handle multiple users at a time using any of the web browsers.
* Reliability: Database updating should follow transaction processing to avoid data inconsistency.
* Availability: The project could be deployed locally or on a public shared server so it will be available all the time and will be accessible anywhere using the internet.
* Security: Only authorized and registered users will be able to access and make changes in the database or access the personal account.

## Use Cases

The purposes of developing use cases are to describe how a user category known as an actor will interact an application to complete a specific action. They tell a story about how an end-user interacts with the system under a specific set of circumstances [19]. The main components of use case (see Table 2. Use case main components.):

|  |  |
| --- | --- |
|  | An actor is a role that a user or external system plays with respect to the system. |
|  | A Use Case captures some functionality of the system. It shows what capabilities will be provided by the system to an actor. |

Table 4. Use case main components.

(see) illustrates the use case of how the admin of the gym club will interact with the system. The main actor involved in this use case is gym administrator who will need to provide a valid username and password in order to log into the system. If the login confirms that an actor is entitled to use the system, then he will be provided with a view of actions he is able to perform (see ):

###################### use case diagram goes here

The System boundary would be a container with all the use cases. There are more use cases to display, only most important are shown. The purpose of these diagrams and use case tables was to illustrate how actors (administrator, members, gym’s IT system) interact with the system, and to show the functionality.

|  |  |
| --- | --- |
| Use case name | Check-in members at the entrance. |
| Actor | Gym Administrator. |
| Description | The administrator finds the member by typing his/her name or ID into the search field to perform check-in. ‘To Pay’ field indicates the amount if the member has to pay before enter. |
| Preconditions | The administrator must be logged-in and the searched member must have an account registered in the database. |
| Trigger | The administrator checks-in the attended member, the check-in time and payment stored into database. |

Table 5. Check-in members at the entrance.

|  |  |
| --- | --- |
| Use case name | Add update member account. |
| Actor | Administrator. |
| Description | The administrator enters the necessary details: first name, last name, email, etc., and selects the membership plan. |
| Preconditions | The administrator must be logged-in and the new member has provided the details and has selected the programme/membership plan. |
| Trigger | The administrator submits new member account into database. |

Table 6. Add update member account.

|  |  |
| --- | --- |
| Use case name | Apply discounts to the membership plans. |
| Actor | Administrator. |
| Description | The Administrator applies certain discounts to membership plans and sets the start date and valid until date.  The discounts can be auto-applied for the particular valued members when their memberships due to expire. |
| Preconditions | The administrator must be logged-in.  The system is set-up determine the expiring memberships. |
| Trigger | The members of the gym are notified about certain discounts via email or via web site’s personal account. |

Table 7. Apply discounts for the membership plan.

|  |  |
| --- | --- |
| Use case name | Apply discounts to the products. |
| Actor | Administrator. |
| Description | The administrator applies certain discounts to the products and sets the start date and valid until date.  The discounts can be auto-applied to particular members if they buy something, get next product with certain discount. |
| Preconditions | The administrator must be logged-in.  The system is set-up to auto-apply discount of the next purchase within particular period of time. |
| Trigger | The members of the gym are notified about certain discounts via email or via web site’s personal account. |

Table 8. Apply discounts to the products.

The use case diagram below (see ) illustrates the use case of how the member of the gym club will interact with the web site provided by the gym club. The main actor involved in this use case is registered member and has an account with the gym club, and who will need to provide a valid username and password in order to log into the web site. If the login confirms that an actor is entitled to login, then he will be provided with a view of actions he is able to perform (see ):

################ use case diagram goes here

|  |  |
| --- | --- |
| Use case name | View available products. |
| Actor | Gym member |
| Description | The member of the gym sees available products at the gym’s point-of-sale including discounts available to particular products. |
| Preconditions | The actor must have a web browser and an internet connection. |
| Trigger | The actor wants to see available products and their discounts if available. |

Table 9. View available products.

|  |  |
| --- | --- |
| Use case diagram | Update personal account details. |
| Actor | Gym member |
| Description | The member of the gym can update his/her personal details, including name, email, address, upload new picture, etc., |
| Preconditions | The actor must be logged-in to the web site’s personal account. |
| Trigger | The actor wants to change his/her personal details. |

Table 10. Update personal account details.

|  |  |
| --- | --- |
| Use case diagram | Check the history of personal account. |
| Actor | Gym member. |
| Description | The member of the gym can check his/her personal account history including the membership plans purchased, the amount spent, due payments, attendance, etc. |
| Preconditions | The actor must be logged-in to the web site’s personal account. |
| Trigger | The actor wants to be advised about the history of the personal account. |

Table 11. Check the history of personal account.

|  |  |
| --- | --- |
| Use case diagram | Reserve the product. |
| Actor | Gym member. |
| Description | The member of the gym can reserve the product available at the point-of-sale and then pay locally at the point-of-sale at the gym club. |
| Preconditions | The actor must be logged-in to the web site’s personal account. |
| Trigger | The actor wants to reserve the product available and to pay at the point-of-sale. |

Table 12. Insert the product.

## Class Diagrams

The class diagram is static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application [20].

The UML diagrams like activity diagram, sequence diagram (latter will be show and described in the next section) can give the sequence flow of the application but class diagram is a bit different.

I have used the class diagram (see ) to model the static view of relationships between the different model classes. The diagram depicts the associative relationships that exists amongst some of the classes. An associative relationship is the most common form of relationship in a class diagram and is shown with a line and arrow and the multiplicity of the association denoted by the numbers stating how many objects that can participate in the relationship. Classes are composed of the three things: a class name, attributes and operations.

######### class diagram goes here

## Communication REST API and Sequence Diagrams

REST works by allowing a method to be called via a HTTP request. In order to facilitate communications a REST API will be used.

Sequence diagrams are created using the events that have been identified by examining a use case. The sequence diagram illustrates how events cause flow from one object to another [19]. The main purpose of a sequence diagram is to define event sequences that result in some desired outcome. The focus is less on messages themselves and more on the order in which messages occur. Most sequence diagrams will communicate what messages are sent between a system’s objects as well as the order in which they occur.

Below (see ) is a sequence diagram illustrating the administrator who wants to retrieve the information about a particular member. The information will contain name, email, address, etc. membership history, expired and current membership plans, due payments, amounts spent on membership plans and various products, etc.

When drawing a sequence diagram, lifeline notation elements I have placed across the top of the diagram. These lifelines represent object instances that participate in the sequence being modelled.

In the example shown (see ) the first lifeline represents an instance of class Admin, whose instance name is admin. When logged on, the admin sends a HTTP request message which represents a ‘getMemberProfile’ method that the receiving ‘MemberController’ class implements. Then ‘MemberController’ object makes a call to the ‘memberService’ object which is an instance of the ‘MemberServiceImplementation’ class. The ‘MemberController’ object is calling the ‘memberService’ object’s ‘getMemberProfile’ method. The ‘memberService’ object then calls ‘getMemberProfile’ method of ‘memberDAO’ object.

The data will be returned as JSON and rendered to user.

Besides just showing message calls on the sequence diagram, the return messages are included. The return messages are an optional part of a sequence diagram. The use of return messages depends on the level of detail/abstraction that is being modelled. I personally like to include return messages whenever a value will be returned, because I find the extra details make a sequence diagram easier to read.

######### getMemberProfile sequence diagram goes here

## Flowcharts

Sometimes it’s more effective to visualize something graphically than it is to describe it with words. That is essence of what flowcharts did for me. Flowcharts explained some processes clearly through symbols and text. Moreover, flowcharts gave me the gist of the process flow in a single glance. The following flowchart diagrams are some graphical explanations of a few processes included in my project:

* The following process model (see ) represents the decision flow made by the application when new member signs up and selects ‘Pay as You Go’ status. The application business logic will detect and act appropriately in regards of further member actions.

########### the new member sign up flowchart goes here

* The following process model (see ) represents the activities when the membership plan is due to expire within next 15 days. The application acts appropriately according how member has been classified, very frequent, frequent or less frequent, and whether member accepts or not the discounts proposed.

########### membership is expiring in 15 days. Flowchart.

* The following process model (see ) represents the actions taken by the software after the member of the gym buys any product at the point-of-sale at the Gym.

########### member buys the product. Flowchart.

## Database Design

The following section describes the database design. As I have had two modules about databases in the second and third year of studies, especially the latter one ‘Advanced Databases’ provided me with kind of sufficient knowledge about database design before I start my fourth year final project. However, this familiarity could have not guaranteed that my database design will be proper. Online research and the book proved to be valuable resources.

Database design is very important to the efficiency of a well running application. If a database is not designed accurately it can slow down the performance of an application. This is process that is used to define and then specify the structure of the objects within a database system. Normalization and Entity Relationship Modelling are two important design aids that can be used to design databases [21].

All functions within the application, from simply logging in, to gathering all history information relies on a database to query and hold the information. This data is stored using MySQL as the database management system.

The entity-relationship (ER) data model allowed me to describe the data involved in my application in terms of objects and their relationships. The ER model is important primarily for its role in database design. It provided useful concepts that allowed me to move from an informal description of what users want from their database to a more detailed and precise description that can be implemented in a DBMS.

Entity relationship modelling means identifying the things (entities) about which information will be stored, their properties and the relationships among them. Chen first introduced the entity relationship model in 1976 [21]. A relationship is an association that exists between one or more entities. There are three possible relationships that can exists between two entities. These are referred to as cardinality and are as follows:

* One-to-one
* One-to-many (or many-to-one)
* Many-to-many

Each entity then needs a primary key to be selected from the fields of that entity whose values uniquely identify one record in a file. A foreign key is key in one table which points to a primary key in another table and provides a link between data in two tables. It acts as cross-reference between tables.

####### database diagram goes here

The Entity Relationship Diagram (see ) shows the entities that have been identified for the Gym Management System and are as follows:

* admin
* members
* membership\_status
* member\_attendance
* programmes\_prices
* products
* products\_sales
* products\_reserved
* offers

The Entity Relationship Diagram (see ) illustrates that the following relationships exists among entities:

* A member can have one or more records in membership\_status table related to membership plans (expired and current)
* A member can have one or more records in member\_attendance table
* A member can have one or more products bought at the point-of-sale
* A member can have one or more products reserved at the point-of-sale
* Many members can have many offers related to membership plans and products at the point-of-sale
* One programme price can have one offer at the same time

# Implementation

## Introduction

In this chapter I will cover the technical implementation as well as the visual components

## Architecture

Spring framework makes the development of web applications very easy by providing the Spring MVC module. Spring MVC module is based on two most popular design patterns – Front controller and MVC [23].

In my opinion, the Spring MVC . Spring’s MVC module is based on front controller design pattern followed by MVC design pattern (see ). All incoming requests are handled by the single servlet named ‘DispatcherServlet’ which acts as the *front controller* in Spring’s MVC module. The ‘DispatcherServlet’ then refers to the ‘HandlerMapping’ to find a controller object which can handle the request. ‘DispatcherServlet’ then dispatches the request to the controller object so that it can actually perform business logic to fulfil the user request.

######## spring mvc architecture here

As Spring have started supporting RESTful web services after Spring 3 version, I chose to follow and concatenate these two technologies for handling all user’s requests. My opinion is good use main business logic for both. As every system needs to use resources

## Directory Structure

In this section, I will describe the applications layout of the Gym Management System and gym’ web site. The main application consists of one main directory called ‘Degree\_Project’ with a number of subdirectories and files. The directory structure of the project can be seen below (see ):

########## project structure goes here. From sts

All actual JSP pages (authentication,jsp, index.jsp, dashboard.jsp, communication.jsp, charts.jsp, products\_sales.jsp, reports.jsp, reports\_big\_table.jsp, settings.jsp, to\_do\_list.jsp) are in the ‘pages’ directory.

Table below (see Table 13. Directory structure) describes the purpose of each of the subdirectories that are contained within the ‘Degree\_Project’ directory:

|  |  |
| --- | --- |
| com/tomas/model | Classes in model layer encapsulates the application data and in general they will consist of POJO |
| com/tomas/controller | Classes in controller layer accepts HTTP requests and loads or save data and returns a response. The response could be a redirect, or a view, or JSON, or String in my application.  Classes in this layer will have access only to service layer classes through interfaces |
| com/tomas/service | Contains classes for service layer. Here I put my business logic. Classes in this layer only have access to others classes in this layer and for specific set of dao layer classes through interfaces. |
| com/tomas/dao | Contains classes that provides CRUD logic for my application (insert, update, delete, select) |
| com/tomas/tests | Contains the test class used for transactional unit test performed during development. |
| resources/css | Contains the style sheets files and used throughout the system |
| resources/images | Contains the images for members profiles, products images, and various images for user interface |
| resources/javascript | Contains the JavaScript files used by the system |

Table 13. Directory structure

### Required Configuration

The ‘/WEB-INF/web.xml’ (see ) file is used to define how to deploy the web module to a Servlet Tomcat container described previously. Basically, ‘web.xml’ tell container all servlets in the web application with ‘<servlet>’ element, then tell container when to use which servlet by the url mapping with ‘<servlet-mapping>’ element.

###### web.xml file

In Spring MVC project, for most cases, the same as in mine, there is only one servlet ‘org.springframework.web.servlet.DispatcherServlet’. This servlet will handle all the requests in my main application. After container hands over the HTTP request to ‘DispatcherServlet’, a controller class will be chosen to handle the request according to ‘@RequestMapping’. The ‘@RequestMapping’ annotation is used to map a URL to either an entire class or a particular handler method (see ).

###### @RequestMapping screenshot goes here

Application Context is the context initialized by ‘ContextLoaderListener’ that I defined in my ‘web.xml’ file (see ). In this configuration I am asking Spring to load ‘spring-servlet.xml’ and create an Application Context from it.

#### spring-servlet.xml file

‘spring-servlet.xml’ file called this way, because I mapped the ‘DispatcherServlet’ in ‘web.xml’ with the servlet-name ‘servlet’. This is by convention. This file is the entry point for Spring, when the ‘DispatcherServlet’ is loaded it finds this file and starts to load its components. This file contains beans elements – the most basic configuration unit in Spring. It tells Spring to create an objects. The ‘id’ attribute of the bean element gives the bean a name by which it will be referred to in the container. A ‘DataSource’ bean is declared for database connection in this file.

## 

## Securing access to the Application

It is very important to take security into consideration when developing a web-based application because it is the part of the Internet and is vulnerable to attack. First, it is important to restrict access to the application by requesting that a user must login to the application to gain access (see Figure 14. Login screen)

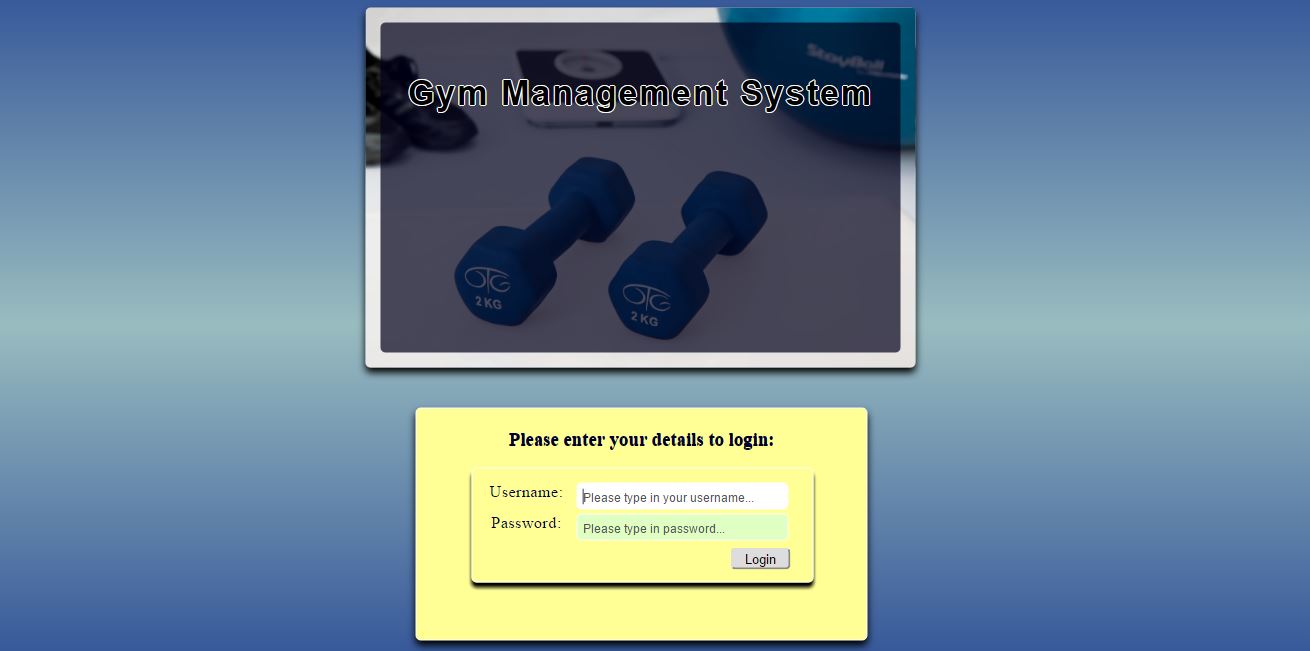


Figure 14. Login screen

The home page of the main application is index.jsp and in order to render this page in your browser you must provide your details like a username and password in authentication.jsp page, then the request will sent to the server and authentication process will be performed.

Thanks to Spring Security [10] I used to intercept sensitive URL requests (like admin or the member will not be able to access application content that are only designed for using by authenticated users). Again, this feature was new to me and I feel significantly improved my development skills, knowing that security against unauthenticated users is vital of this type of web applications.

At a very high level, spring security is the servlet filter (see ) which intercepts every request to the server side. Then, depending on how it was configured it will either allow to proceed through or will send back a notation about authentication failure.

######## security spring from udemy diagram

Some security concepts:

* **Authentication:** the process by which users tell the application who they are
* **Authorization:** is the set of roles that users assume when they authenticate and define what they can or can’t see and do (this applies to the gym’s website)

To set up spring security for my application I provided

This is very useful feature of Spring where I was able to integrate Spring Security with a Spring MVC framework. This feature gave ability to encode the login passwords for the admin and members and store it into the database.

Saving your password in a simple string form in database is never a good idea, instead one must save his password in some sort of encoded or hashed format. To make password purely protected we must have some algorithm to identify how to save encoded password in database and how to authenticate user using an encoded password stored in database.

To encode a password using 'BCryptPasswordEncoder'[14] spring-security provides ‘encode()’ function of 'BCryptPasswordEncoder'class (see Figure 9):

When I start creating the gym’s website it is my intension was to add authorization and authentication support also, and I had to decide which parts of the web site will be public accessible and which parts will be required users to login to access them.

# Evaluation

## Testing

This chapter explains the testing and validation of implemented software, at the end of which, a working system should be ready for delivery to users.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is a set of activities that can be planned in advance and conducted systematically.

For this reason a template for software testing a set of steps into which we can place specific test case design techniques and testing methods should be defined for the software process.

The testing process involved a number of different techniques and tools, in this chapter I will cover the main ones used, followed by the details of the actual user testing that was performed.

Testing is carried in two phases. First phase is carried during the software engineering, development, during the module creation. Second phase is after the completion of software. This is system testing which verifies that the whole set of programs working together.

In the first phase testing process should receive some inputs, processes them, and provides some outputs. It includes several activities, which should be carried out in different stages of software development. With the testing I was focusing on the possible unexpected outputs. This way I could correct the code as I went.

## Unit testing

Unit testing was performed to ensure smaller components of the application performed according to specification, this would then be followed by a level of integration testing to ensure the individual components worked together as expected.

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. The unit test is white-box oriented, and the step can be conducted in parallel for multiple components.

## Unit integration testing

This testing is performed on the program’s structure when new modules being added while at the same time constructing test to uncover errors associated with interfacing the modules.

* Interfaces for the modules should be tested in the way that these interfaces perform their functions and make sure that they do what they are expected to do and also they do not do anything else which would lead to breaching of object-oriented rules.

Spring is a great framework to develop enterprise Java web applications. It provides a lot of features and one of them is the **TestContext** Framework, which helps us to implement integration unit tests easily in our applications [5].

I have created a model class with name **Member**. It has the following attributes, **firstName, lastName, address, email, dateOfBirth, email, password**, and accessor methods for them.

I also created **Service** and **DAO** classes as follows, in order to perform persistence operations with my domain model.

**MemberDAO** is an interface which defines persistence operation over **Member** instances like **addMember, memberProfile, updateMember, deleteMember**.

**MemberDAOImplementation** is an implementation of **MemberDAO** interface which employs **NamedParameterJbdcTemplate** bean of Spring in order to implement persistence operations via JDBC API [6].

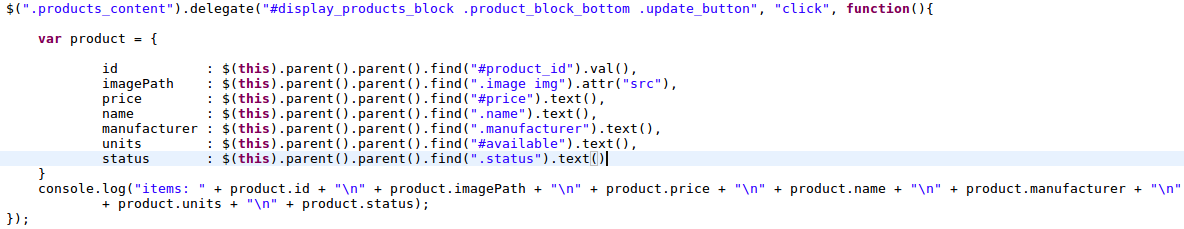
Then I wrote a transactional integration unit test where I have included test methods to test adding new Member instance, deletion of an existing one and finding by its id.

**@RunWith** annotation belongs to Junit, and is used to tell my STS IDE which Runner class, SpringJUnit4ClassRunner.class in this case, to use to run test methods defined in the class.

Therefore, SpringJUnit4ClassRunner starts a new transaction at the beginning of each test method execution, and then rolls back it at the end. The reason to rollback instead of commit is that those changes performed on the database within each test method should not adversely affect execution of other integration tests. However, any service method call which expects an active transaction to work during its execution is satisfied with that active transaction spanning the test method. It is possible to see how transaction is created and then rolled back from the log messages shown below.

Finally I have run the tests with JUnit and observed the results. The Junit tests passed as follows.

## Validation testing





## Performance testing

Performance testing is designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Even at the unit level, the performance of an individual module may be assessed as white-box tests are conducted. However, it is not until all system elements are fully integrated that the true performance of a system can be ascertained.

* Data collection technique: stop watch. For this it’s important to have a clear definition of when a task starts and stops
* Time on task - Time taken to complete a task
* Number of tasks completed in a given time
* Time spent recovering from errors. Number of errors. Number of subsequent moves to rectify error
* Number of clicks required for task

## User testing

After I have developed my application, it got to a point where I felt the application was ready, I felt confident I had tested all scenarios and had fail safes in place to handle all situations.

While I was indeed confident with my application, a level of actual user testing would be required. I felt I knew how the application would and should work, however it was unlikely I had tested all scenarios, even if the application was bug free, it might have still been rejected by users due to other reasons.

To get know my application rating evaluated, along with finding outstanding bugs, I enlisted the help of some family and friends. I selected 7 people, each were given a link to the my application which is hosted in the red hat cloud provider named as ‘Openshift’ at: <http://gym-tomas111.rhcloud.com>. Each person was asked to complete a set tasks provided by me, evaluate different features that I have, just to talk which of them most successful comparing with the time that I put for work implementing them and which of them the users liked the most, so I can put their feedback in the document and measure them, i.e. *‘I found this really easy to use, or I found the layout very good, I found it very functional*’.

* How did you find the registration/add new member process?
* How easy was to accomplish basic tasks the first time you encounter the design?
* Did you find the user interface easy to understand?
* How would you rate the effectiveness of the application?
* Would you recommend this application to others?

Each participant was also asked to make a note of anything they didn’t like or found difficult to understand.

At the end I have asked users to evaluate

# Conclusion

* Learn the Apache Tomcat web server working functionalities, application deployment and managing the lifecycle of the application within the server
* Learn the Spring and Spring MVC framework, which is the most popular application development framework for enterprise Java
* Learn the JDBC Template class, which simplifies the use of JDBC and helps to avoid common errors.
* Deepen my knowledge and understanding of Java programming language, also JavaScript, JQUERY, AJAX technologies, HTML, CSS

# References

1. GYMMASTER [Online]. Available from: <https://www.gymmastersoftware.com/> [Accessed on: 04 November 2015].
2. CAPTERRA TECHNOLOGY BLOG [Online]. Available from: <http://blog.capterra.com/top-7-free-open-source-membership-management-software-products/> [Accessed on: 04 November 2015].
3. DUCKET, J. (2013) *HTML and CSS: Design and Build Websites*. 1st Ed. Published by Wiley, Inc.
4. SUEHRING, S. (2013) *JavaScript step by step*. Third Ed. O’Reilly Media, Inc.
5. MCFARLAND, D. S. (2012) *JavaScript and JQuery: the missing manual*. Second Ed. O’Reilly Media, Inc.
6. From Wikipedia. *Database Connection*. Available from: <https://en.wikipedia.org/wiki/Database_connection/> [Accessed: 03 November 2015].
7. From Spring.docs.io. *Integration testing*. Available from: <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/integration-testing.html> [Accessed: 05 September 2016].
8. From Java Code Geeks. *Unit integration tests*. Available from: <https://examples.javacodegeeks.com/enterprise-java/spring/write-transactional-unit-tests-spring/> [Accessed: 05 September 2016].

# References

|  |  |
| --- | --- |
| [1] | Statista, “Facts on Health & Fitness Clubs,” [Online]. Available: https://www.statista.com/topics/1141/health-and-fitness-clubs/. [Accessed 01 12 2015]. |
| [2] | Capterra, “Open Source Membership Management Software Products,” 21 10 2016. [Online]. Available: http://blog.capterra.com/top-7-free-open-source-membership-management-software-products/. |
| [3] | Q. T., Visual Modelling with Rational Rose 2002 and UML, 1th ed., Addison Wesley, 2003. |
| [4] | B. L. D. Whitten J. L., Systems Analysis and Design Methods, McGraw Hill, 2005. |
| [5] | Lucidchart, “Free Flowchart Maker,” [Online]. Available: https://www.lucidchart.com/pages/landing/free-flowchart-maker. [Accessed 20 08 2016]. |
| [6] | D. JOHN, HTML and CSS: Design and Build Websites, 1st Ed. ed., Wiley, Inc., 2013. |
| [7] | S. S., JavaScript step by step, Third ed., O'Reilly Media, Inc., 2013. |
| [8] | M. D., JavaScript and JQUERY: the missing manual, Second ed., O'Reilly Media, Inc., 2012. |
| [9] | A. R. SCHUTTA NATHANIEL, Foundations of Ajax, 1st. ed., Apress, 2006. |
| [10] | W. CRAIG, Spring in Action, Third ed., Manning Publications Co., 2011. |
| [11] | A. T. 8, “Apache Tomcat 8,” [Online]. Available: https://tomcat.apache.org/tomcat-8.0-doc/introduction.html. [Accessed 01 06 2016]. |
| [12] | M. Tutorial, “MySql Tutorial,” [Online]. Available: http://www.mysqltutorial.org/. [Accessed 01 06 2016]. |
| [13] | Spring, “Spring: easy way of learning,” tutorialspoint.com, 2013. |
| [14] | CodeJava, “CodeJava: Coding your passion,” [Online]. Available: http://www.codejava.net/frameworks/spring. [Accessed 15 07 2016]. |
| [15] | Oracle, “Class Logger,” [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/util/logging/Logger.html. [Accessed 20 08 2016]. |
| [16] | S. b. Pivotal, “Spring Tool Suite Downloads,” [Online]. Available: https://spring.io/tools/sts/all. [Accessed 20 04 2016]. |
| [17] | Apache, “The Apache Software Foundation,” [Online]. Available: https://www.apache.org/. [Accessed 01 06 2016]. |
| [18] | R. S. Richardson L., RESTful Web Services, O'Reilly, 2007. |
| [19] | P. R. S., Software Engineering a Practioner's Approach, 6th ed., McGraw-Hill, 2005. |
| [20] | Tutorialspoint, “UML - Class Diagram,” [Online]. Available: https://www.tutorialspoint.com/uml/uml\_class\_diagram.htm. [Accessed 20 08 2016]. |
| [21] | N. S. Elmasri R., Fundamentals of Database System, 3th ed., Addison Wesley, 2000. |

# Appendices

(Listed from A to Z