

100110surfingfish1010100

**“SURFINGFISH  
(WEB AND GSM ENABLED  
AQUARIUM CONTROL SYSTEM WITH SOCIAL NETWORKING  
FEATURES)”  
IN  
EMO PROJECT COMPETITION**

**Project Supervisor: Assist. Prof. Dr Osman Kaan Erol**

**Project Team from KADIR HAS UNIVERSITY**

Ahmet ARDAL	EE
Çağrı İLBAN	EE
Ender PIYALE	EE

**Faculty of Engineering  
Kadir Has University**

**May 2009**

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS	4
ABSTRACT	5
1 INTRODUCTION	6
1.1 Instruction	6
1.2 Motivation	6
1.3 Objective	7
2 APPROACH	8
2.1 Identification of Problem	8
2.2 Classification of Problem	8
2.3 Problem 5H 1W	8
2.4 Fishbone Diagram	8
3 METHOD	9
3.1 Hardware	9
3.1.1 Project Pier	9
3.1.2 Power Supply	10
3.1.3 PIC24FJ96GA008	11
3.1.4 User Interfacing on Hardware	12
3.1.5 Relay Control	14
3.1.6 Filtering	15
3.1.7 Temperature Control	15
3.1.7.1 LM35A Temperature sensor	17
3.1.7.2 Hysteresis Control	18
3.1.8 Lighting	18
3.1.9 Water Level Monitor	18
3.1.10 pH Control	19
3.1.11 Feeding	19
3.1.12 RTC (Real Time Clock)	21
3.1.13 Language Option	22
3.1.14 Ethernet Module	22
3.1.15 System Box	23

3.2 Web	27
3.2.1 Architectural Overview	27
3.2.1.1 Technology Overview: ASP.NET Web Application	27
3.2.1.2 Technology Overview : AJAX	32
3.2.1.3 Technology Overview : MySQL(Database)	35
3.2.1.4 Software Overview: "Whidbey" Visual Studio.NET (IDE)	35
3.2.1.5 Software Overview: EMS SQL Manager For MySQL	35
3.2.2 The 3-Tier approach	37
3.2.3 The Database	38
3.2.3.1 Tables	39
3.2.3.2 Stored Procedures	43
3.2.4 DAL – Data Access Layer	49
3.2.4.1 DAL introduction	49
3.2.4.2 DAL classes	49
3.2.5 The Business Logic Layer	50
3.2.5.1 Classes	52
3.2.6 Presentation Layer	53
3.2.6.1 XHTML & CSS	53
3.2.6.2 Pages	53
4 PRICE LIST	58
5 CONCLUSION	59
6 TABLE OF FIGURES & TABLES	60
7 REFERENCES	63
8 CONTACTS	64

## **ACKNOWLEDGEMENTS**

We would like to deeply thank to Assist.Prof.Dr.Osman Kaan EROL for his invaluable guidance, kindness and understanding during this project. Without his care and consideration, this project would likely not have matured.

## ABSTRACT

Proper aquarium maintenance requires attention and significant amount of time. A specific knowledge about fishes and their requirements should be known. Aquarist should be able to operate various kinds of maintenance tools such as filters, motors etc... This project presents a microcontroller based solution to assist aquarist for important aquarium maintenance issues.

Primary, problem statement is done and targets of project are declared. The solutions, which are offered for “autonomous” control of aquarium maintenance, are briefed and systems are described.

Project team intended to design and implement a system which is capable to perform followings according to demands of aquarist: adjustment of water temperature, feeding, lighting, water level monitor, and filtering.

And main issue is “socializing standard aquariums”. Project includes web utility. Social networking tool on the Web that connects aquarists worldwide and it manages sharing aquarium datas between aquarists over web.

## 1. INTRODUCTION

### 1.1. Instruction

Having an aquarium should be considered as more than a hobby. Generally, it is presumed as putting some fish into a water tank and watching joyfully. However, inattentive and insufficient attempts, which result in to bad experiences, would show the reality: Having an aquarium requires many strict maintenance regulations according to the dimensions, fishes and many other facts. Aquarium imitates the “real life” in a very small place and the aquarist should maintain this by taking many considerations according to changing conditions.

Many fishes originate from trophic climates and their living requirements can be variable compared to aquarium environment. Therefore, controlling aquarium temperature and PH degree is essential for the health of fishes. For instance, South American fishes require acidic (PH 5.5-7) water while African fishes require basic (PH 7-8.5) water[2]. PH degree computation is done by logarithmic calculations and requires continuous monitoring. On the other hand, water temperature should be controlled in order to avoid temperature difference between day and night. Moreover, aquarium should not be lightened more than 8-12 hours[1]. These are only some of main maintenance issues. As it is seen, aquarium maintenance is hard and requires intensive care.

This project presents a microcontroller-based solution to assist aquarist in conducting maintenance. System continuously monitors the aquarium and performs necessary actions by controlling various peripherals.

### 1.2. Motivation

Aquarium maintenance requires specific knowledge and allocation of important amount of time. This system offers an autonomous control for challenging aquarium maintenance issues. It will be helpful for the ones who can not share time for aquarium maintenance or who uses it for decorative purposes.

### **1.3 Objective**

Project team intended to design and implement a system which is capable to perform following according to demands of aquarist: adjustment of water temperature, feeding, lighting, water level monitor and effective filtering.

The system is based on microcontroller which is programmed to perform as an *operating system*. System has an *MIMO* structure where multiple inputs are taken from sensors and processed to give a control output through electric relays. System is capable to perform multiple tasks simultaneously thanks to its program architecture.

Microcontroller programming architecture (Operating System) also permits further modifications such as new languages or user-demanded adjustments. New maintenance tasks can be inserted to programme without changing its “core” and this provides an easy environment for product development. The system contains RS232 port to add new features and user-demanded adjustments to microcontroller. Also, an ICSP(In Circuit Serial Programming) port provides accessibility for high-level programming issues which may concern operating system.

System has an user-friendly and understandable interface to accept user inputs. LCD screen and keypad are selected according to provide ergonomic interface. User can choose either English and Turkish menus or add a new language easily. The location and type of connectors are determined according to safety issues to avoid any electric accident.

**Keywords:** Aquarium, Microcontroller , Operating System, Aquarium Maintenance, Engineering, Ethernet, Tcp/Ip

## 2. APPROACH

### **2.1 Problem Identification**

Aquariums have a big number of lovers and they are preferred as primary decoration item in restaurants and shopping centers. However, the complexity of aquarium maintenance creates various problems such as losing fishes, costly part replacements etc... Generally, the hobby customers leave their aquariums early or the decorative users do not prefer aquariums due to their hard maintenance.

There are existing control systems which intended to help aquarist to conduct maintenance but they are generally insufficient to progress a fully-autonomous control. Some of existing systems may contain autonomous control however their structures are inflexible and can not be adapted or modified easily by aquarist. Other fully-autonomous systems are much above of reasonable prices.

### **2.2 Problem Classification**

The required maintenance issues are certain and the tools(filters, motors, heater etc...) are known. The techniques are determined to maintain an aquarium and this project does not offer to improve them. An automatic control system which purposed to coordinate various peripherals is needed. Therefore;

- Aquarist expectances are certain.
- Maintenance techniques are known.
- Solution is obviously a system which will coordinate peripherals.

A closed-end, design problem.

### **2.3 Problem 5W 1H**

**What ?** : Aquariums require intensive maintenance

**Where ?** : Residences, restaurants, shopping centers etc...

**When ?** : Since aquariums became popular and their prices reduced to acceptable levels.

**Who ?** : Home users(hobby) and business owners(decorative).

**Why ?** : Maintenance variability according to changing environment and kind of fishes.

**How ?** : By causing fish losses, bad smell or bad view in aquarium.

## 3 METHODS

### 3.1 Hardware

#### 3.1.1 Project Pier



Figure 3.1 - Project Pier logo

In our project we decided to use a project tracking software. After looking around for a while we've decided to give Project Pier a try for project tracking. It could be a good choice for us for a number of reasons:

- It's free,
- It's multi-platform
- It's open source and has open APIs (an API is essentially a translator that helps programmers make a program that talks to your program) so if we can't get the devs or community to make a plug-in, I can potentially make it myself or have it made
- It runs on a Web server so we could access it from anywhere

Welcome back Ender PİYALE (Logout),  
Account ▾ Projects ▾

overview my projects my tasks

» Dashboard » My projects

## My projects

**AquaNET**

"AquaNET: Web&GSM-Enabled Aquarium Control System with Social Networking Features" project includes hardware and software implementation of a comprehensive aquarium control system with many sophisticated features such as: Web and GSM connectivity, Rich user interface via a colored TFT screen with a touch panel, PC software for control & monitoring and offline firmware upgrade over USB, Web control panel and online firmware upgrade utility, GSM control & monitoring facility via SMS messages, Social networking tool on the Web that connects AquaNET users worldwide, Standard control & monitoring jobs, such as temperature, water level, feeding, lighting control etc., and many more...

*Involved companies:* SurfingFish Technology  
*Started on:* Dec 06, 2008 | surfingfish

**Active projects**

- AquaNET

Figure 3.2 - Project Pier Page

### 3.1.2 Power Supply

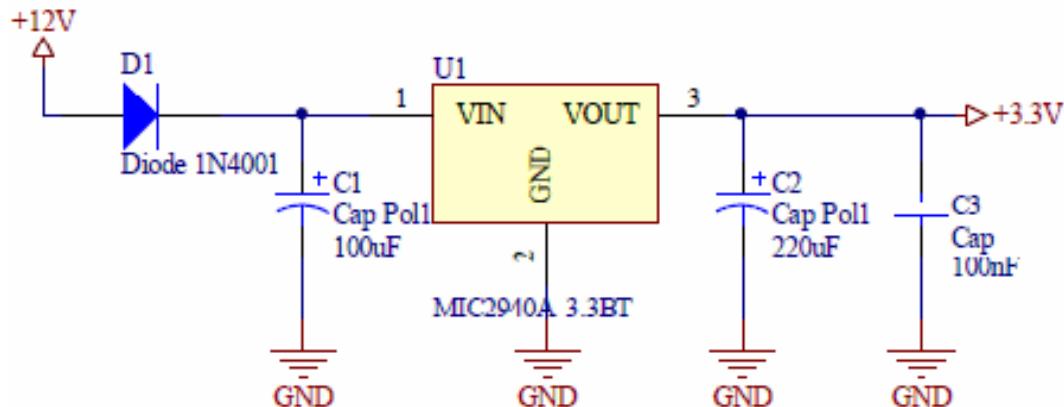


Figure 3.3 - Power supply 3.3V output

We have two main circuits in our aquarium control system and they have power supply's on. We need 12V for cooler fan, uln2003a, lcd, 5V for Rtc, keypad controller and 3.3V for Pic24, ethernet controller etc. So we get 12 V input from adapter and organize that input with linear regulators to 5V and 3.3V. Especially while using ethernet controller, 3.3V regulator MIC2910A needs heat sink because it warms up too much.

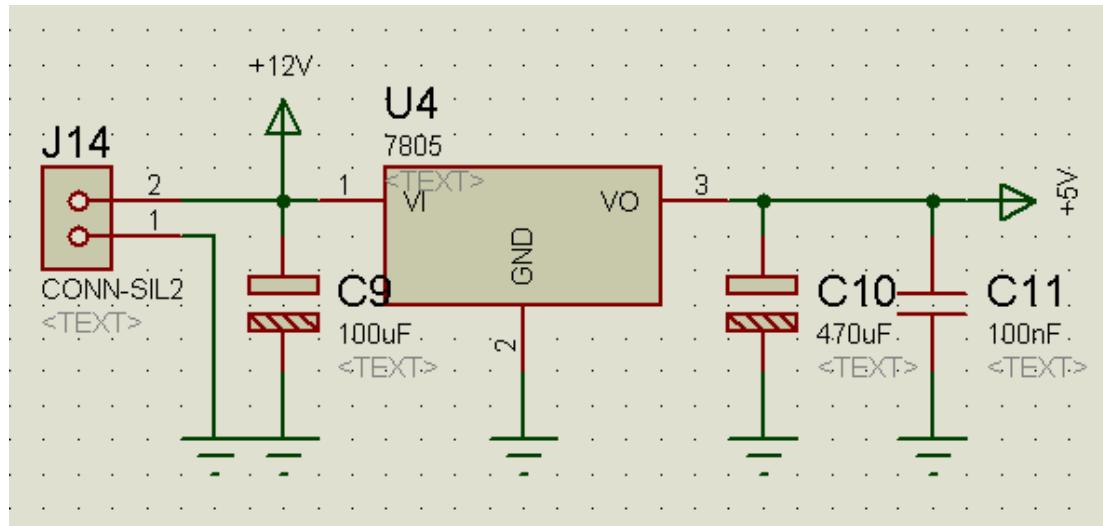


Figure 3.4 - Power supply 5V output

### 3.1.3 PIC24FJ96GA008

Pic 24 is Microchip's new microcontroller family.. We used PIC24FJ96GA008 in this project because of many reasons. We can see some specifications of Pic 24 on picture below.



Figure 3.5 - Pic24 general

That's easy to implement ethernet or usb with this microcontroller, because it's working on 10Mhz and has large program memory which are too important for many applications especially ethernet.

Tcp/Ip stack needs large program memory which are important on embedded systems. So PIC24FJ96GA008 is best choice for this embedded system design project. But this type of microcontroller's are in smd package. So we can't solder it with our hands and then we bought an adapter with PIC24FJ96GA008. We can see it below.



Figure 3.6 – Pic24FJ series

### 3.1.4 User Interfacing on Hardware

We used 4\*20 Lcd and keypad for user interface. User can enter system parameters by keypad and control parameters on Lcd panel.



Figure 3.7 – Lcd Screen

We have to use special integrated circuit (MM74C922) for keypad. This IC inputs keypad and generate 4 bit binary output to the Mcu. And it solves keypad debouncing in it.

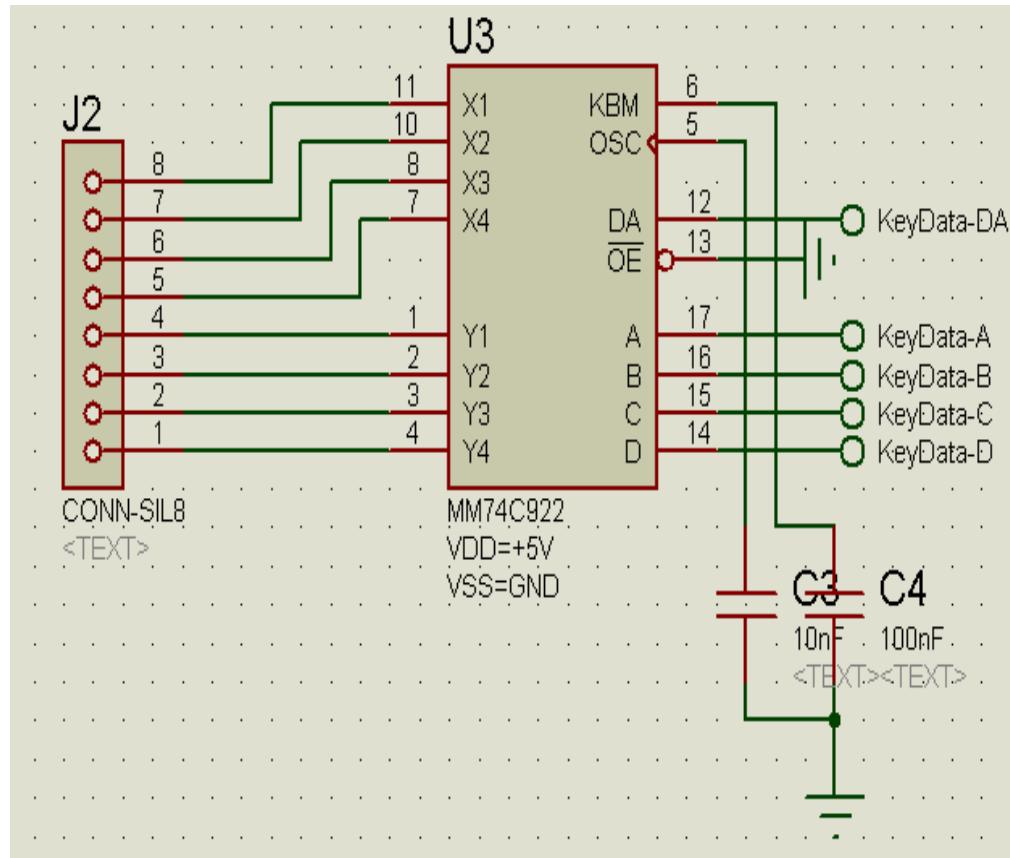


Figure 3.8 – Keypad controller



Figure 3.9 – Keypad interface mechanism

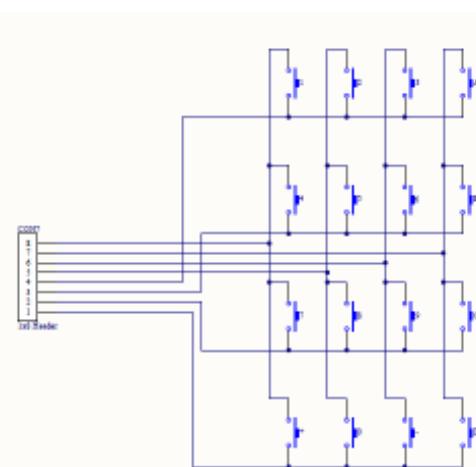


Figure 3.10 – Keypad working

### 3.1.5 Relay Control

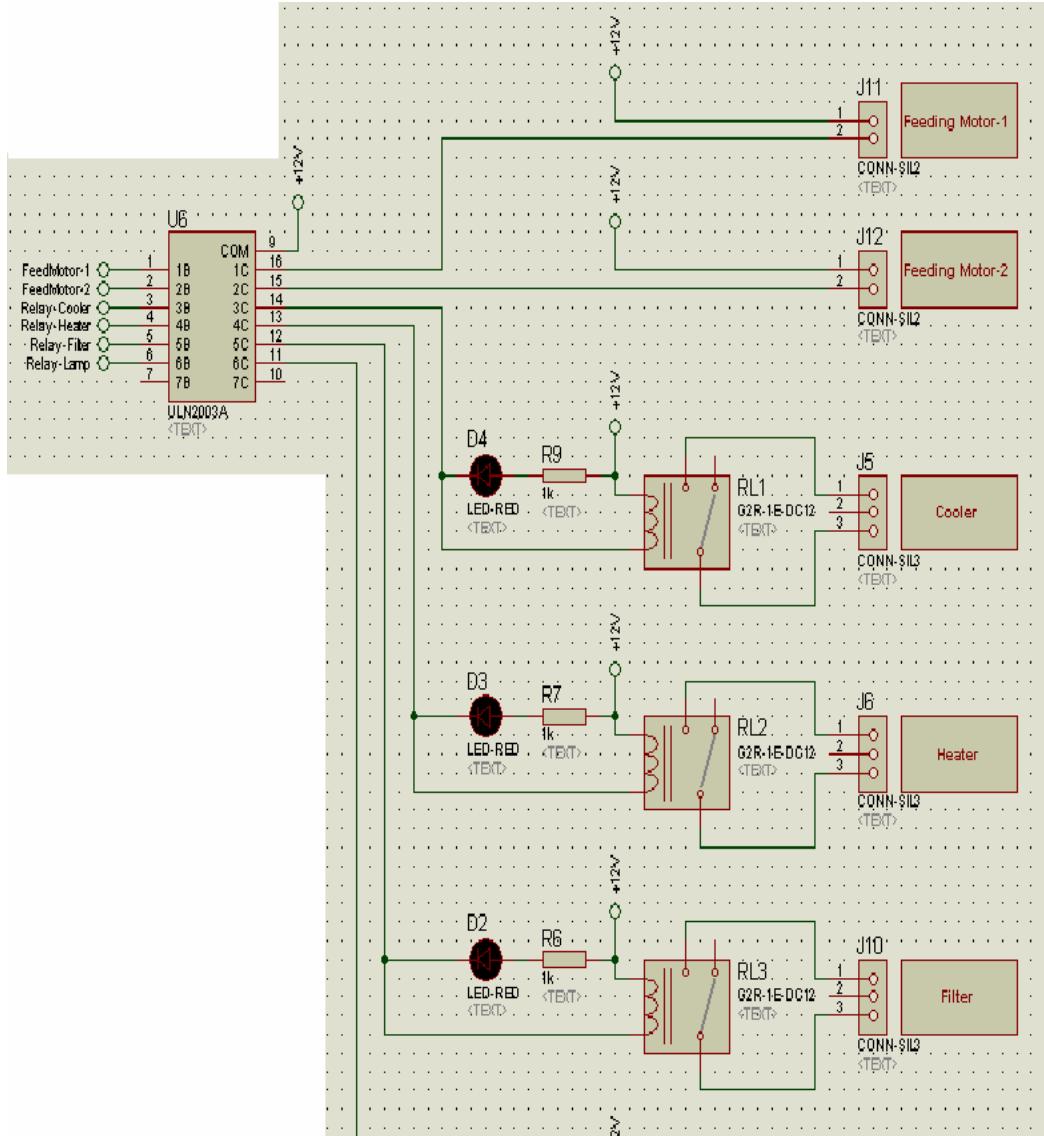


Figure 3.11 – Relays

We use relays to control our outputs like filter, cooler , lights, etc. It's easy to implement and control. We see that on schematic, relays are connected to ULN2003A IC. This is a high voltage high current darlington transistor array. We need this IC because of mcu's output can't supply over 25ma. We must provide 50ma to use relays' inputs.

### 3.1.6 Filtering

Filtering is essential to provide a proper environment for better fishes. Dirts, remains of feeds and harmful wastes should be filtered continuously. Several divided parts of aquarium are dedicated to progress filtering continuously as shown in figure(3.12).

First part provides dirt filtering which is intended to clean rough, big items on aquarium water. An overflow of water results the filtered water to pass to second divided part which contains biological filters where useful bacteria populations to make water very close to real environment. This section contains filters called *BioBall* which provide sufficient and suitable surface area for bacteria population[4]. The last part contains *active carbon filters* where toxics in aquarium water are cleaned.

Cleaned water then comes to main water collecting part where heated or cooled water are also collected. Finally, a motor pumps the cleaned water back to aquarium. Motor is capable to pump 600 lt/h and consumes 12W in optimum operation.

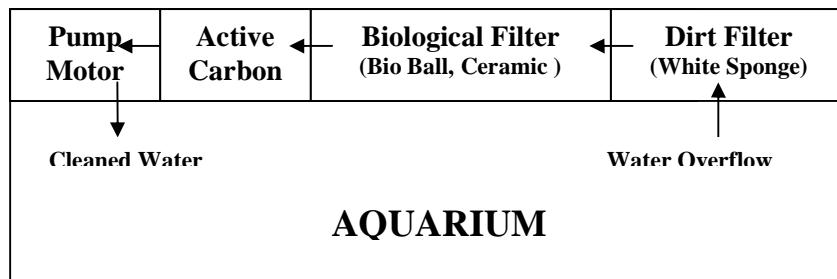


Figure 3.12 - Filtering. Upper view of Aquarium

Filtering is always open except for user-obtained duration to prevent waves on aquarium water while fishes are feeding. The feeding control action deactivates filtering for a certain amount of time.

### 3.1.7 Temperature Control

Most fishes are trophic and their necessities of environment show much difference. Basically, trophic fishes are sensitive to temperature changes[1]. Therefore, aquarium temperature should be monitored and kept under control as shown in fig(3.13).

Temperature difference between day-night and seasonal changes should be taken into account.

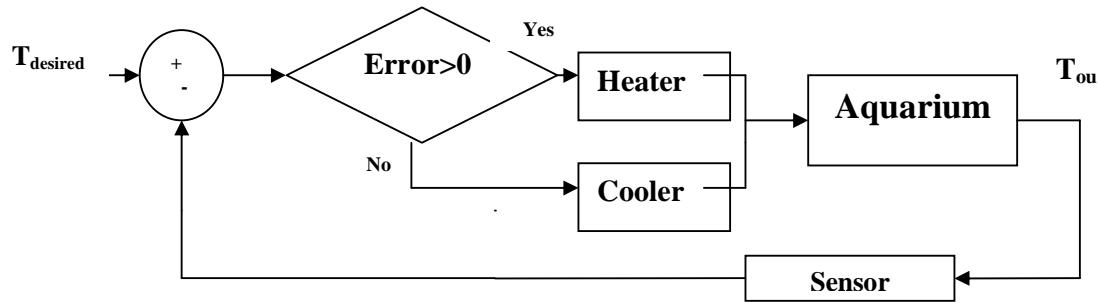


Fig 3.13 - Temperature Control

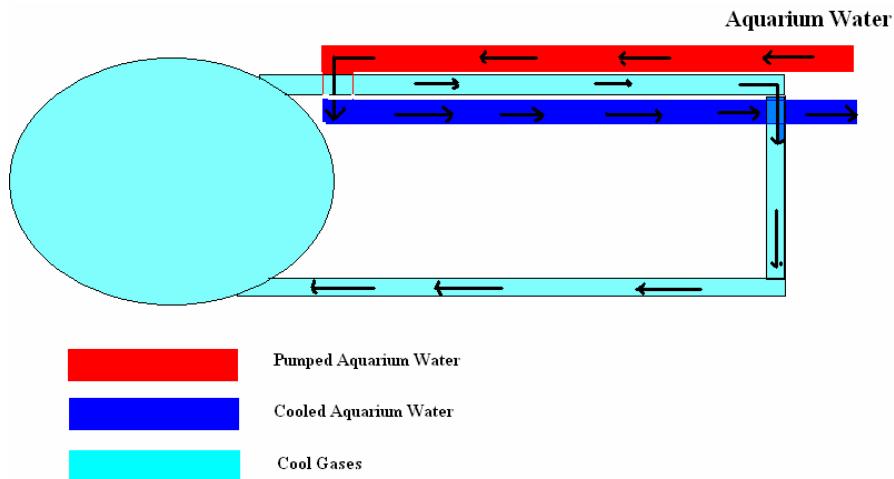
Another important fact about temperature control is the reaction time. The heater/cooler system should be able to rise or decrease the temperature in a reasonable time. The power limitations and aquarium dimensions plays an important role at this point. Project group conducted calculations and experiments in order to obtain best solution. Primary, using a *Peltier* (A thermo-electric device) is considered and following results are obtained as seen in table(3.1).

Table 3.1 - Power VS Required Time to Change a 70 lt. Aquarium's Temperature by 1 C

(Watt)	Seconds	Minutes
45	6508,44	108,47
80	3661	61,016
121	2420,49	40,341
136	2153,52	35,892
168	1743,33	29,055
230	1273,39	21,223
400	732,2	12,203
545	537,394	8,9565

Results show that using *Peltier* for medium size aquariums are impractical according to power limitations of system. Also, *Peltier* requires a power MOSFET driver circuit and this creates adaptation problem when its needed to implement to another system

Therefore, a simpler but effective solution, fridge cooler is considered. Fridge cooler can be driven by relays easily and gives satisfactory performance compared to its price.



*Figure 3.14 - Water Cooler System*

Fridge cooler produces cool gases and circulates in its own system as shown in fig(3.14). A motor, which is located inside aquarium, pumps water into a metal tube where it will be curled to cool gas tube.

### 3.1.7.1 LM35A Temperature sensor



LM35DZ is a analog tempreature sensor. It sense temperature and generate analog output and then we read this with microcontroller. On microcontroller side, we convert analog signal to digital and use that value.

*Figure 3.15 – Temperature sensor*

### **3.1.7.2 Hysteresis Control**

In order to extend parts' life longer (espacially relays), oscillations caused by frequent switching should be avoided. This can be achvied by determining an acceptable interval around user-obtained temperature level. Relays are activated if only upper and lower bounds of desired temperature is exceed.

### **3.1.8 Lighting**

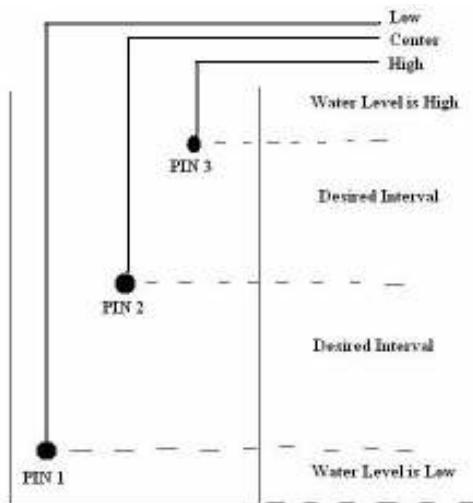
Proper lighting plays two important roles: One is to simulate day-night change in order to provide real life enviroment for fishes[1]. The lighting is automatically turned on and off according to user demands. Another is placed on very critical step of life cycle in aquarium.

The carbondioxide-oxygen cycle is completed by plants leaving in the aquarium. These plants require a specific interval of light spectrum. Therefore, a special type of fluorescent lighting is used which can provide the necessary spectrum for water plants.

### **3.1.9 Water Level Monitor**

Evaporation decreases water level slowly over the time. Keeping water level at certain level is important because water amount is taken as reference in many system calculations. Proper enviroment espacially concerns with water amount.

Water level is sensed by using a probe which is located on desired levels of aquarium glass. System consists of three pins as shown in fig(3.16). The basic idea is: If water level decreases below Pin 2, then Pin 2 and Pin 1 is open-circuited to give alarm to user that water level is below desired height. Also, “water level high” alarm is given when Pin 3 and Pin 1 is short-circuited.



*Figure 3.16 - Water Level Monitor and Possible Alarms*

### 3.1.10 pH Control

pH describes the acidic or basic degree of water. Simply, It gives the concentration of hydrogen ions on water. pH degree of aquarium requires continuous monitoring and control due to fact that fishes can only live in certain pH degrees and they are very sensitive to sudden pH changes[3].

The pH degree of aquarium is measured by a *electronic pH probe* and progressed on microcontroller. Microcontroller compares the measurement with stored(desired) degree and actives the carbon tank according to necessity.

### 3.1.11 Feeding

Microcontroller is capable to plan and distribute feeds to aquarium regularly by simply relaying feed box's dispenser. Filter motor is deactivated during this process in order to avoid waves on water while fishes are feeding.

We design feeding machine but not implement yet. Also our control circuit and software ready for feeding. We can see feeding machines design below.

It's a circular design with too many cell. These cells includes foods for fishes.

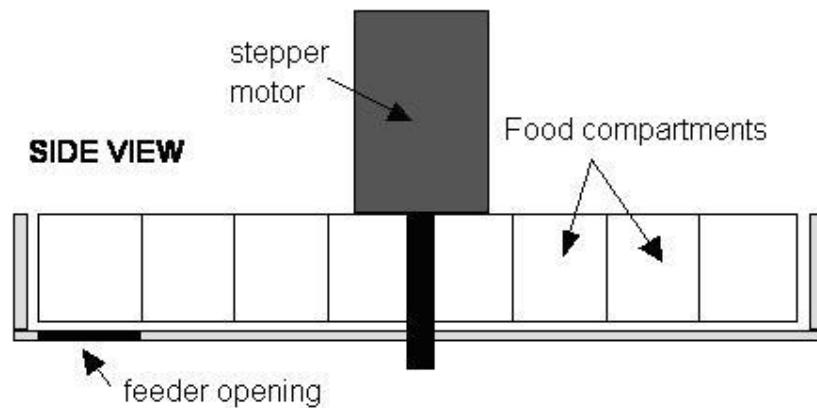


Figure 3.17 – Feeder side view

In this design we have a dc or stepper motor to turn circular part with cells. Black sell on picture shows that there is a hole under the feeding system and every move, foods are felt down through this hole to the aquariums water.

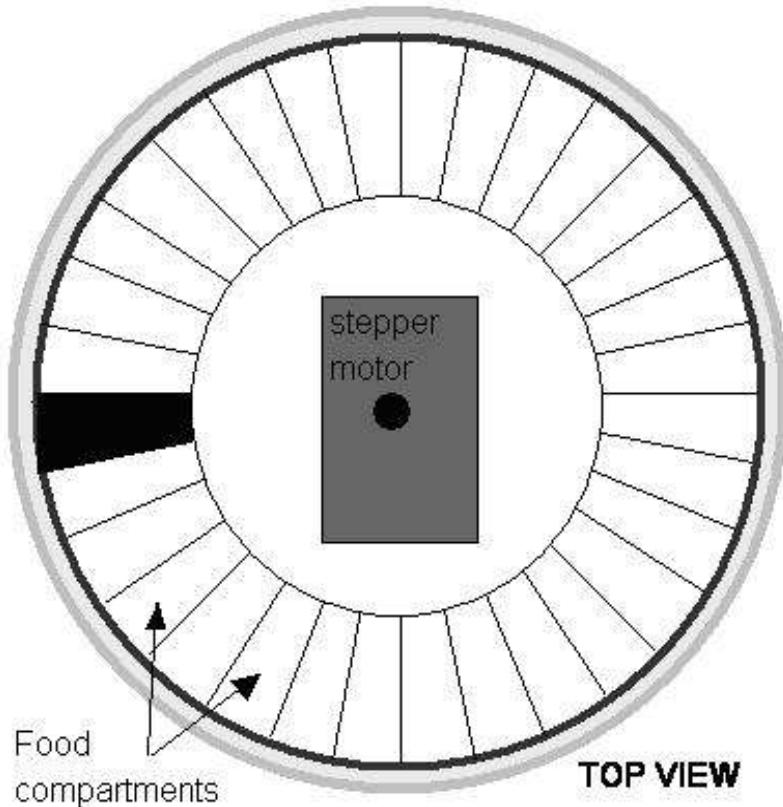


Figure 3.18 – Feeder top view

### 3.1.12 RTC (Real Time Clock)

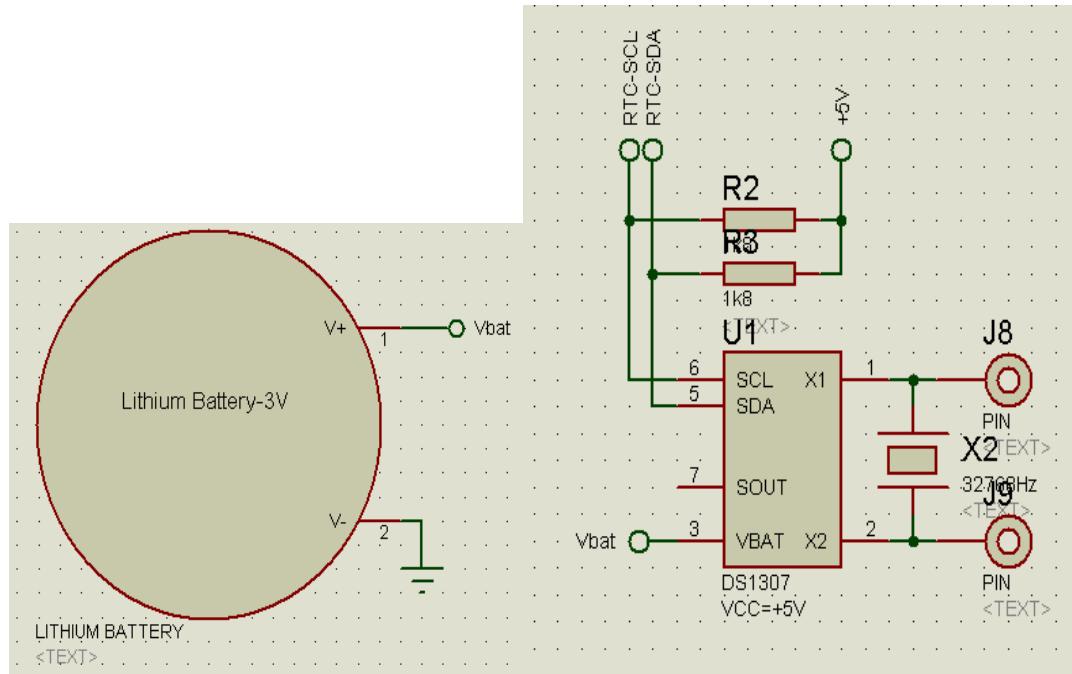


Figure 3.19 – Reel time clock

With Real Time Clock IC, DS1307 we can store real time clock without external voltage need. After programming mcu, you can enter time once in start and it stores time, day, month, year etc. without external power supply up to a hundred year with 3V lithium battery.

So users parameter's which was entered before by aquarist is never lost or mistake.

### 3.1.13 Language option

We use a programming technique to implement special feature. Our system can be used with two language (English and Turkish) at this time but it's easy to add new languages on system.

Aquarist can be chose language with keypad and lcd interface. Also it can be change any time.

### 3.1.14 Ethernet module

We bought Mikroelectronica's ethernet module for our circuit. It has SPI interface with its ethernet controller ENC28J60. Also we can do a printed circuit board for ethernet but we bought it because of time constraint.

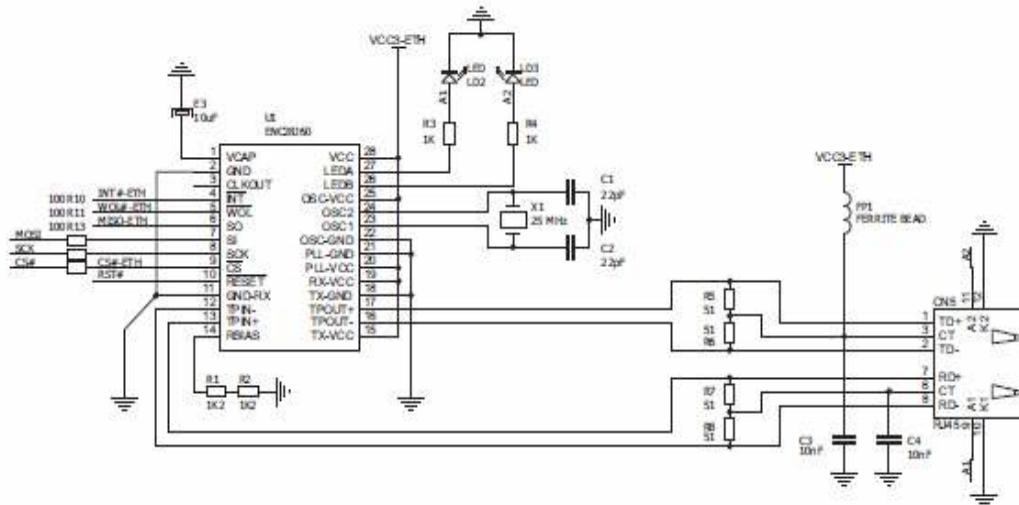


Figure 3.20 – ENC28J60

We used this module because it has 8 Kbytes of buffer ram, SPI serial interface. [5]

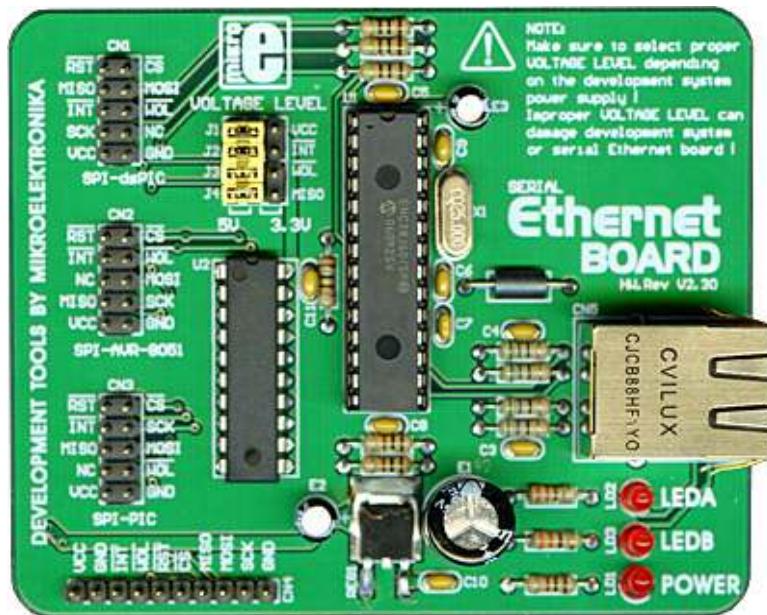


Figure 3.21 – Ethernet board

### 3.1.15 System Box

This system is a version two of our aquarium control project. First version of this project have bad box design because of time constraint. We can see it on picture below. It has jacks on it which was using for controlling heater, cooler, filter motor etc.



Figure 3.22 – Old system box

But in version 2 we made some differences. We changed jacks with another system. We implemented jacks with six input. On original parts inputs connected to 220V directly but we opened and modified it. We connected different cables to the inputs.



Figure 3.23 – First version of jacks

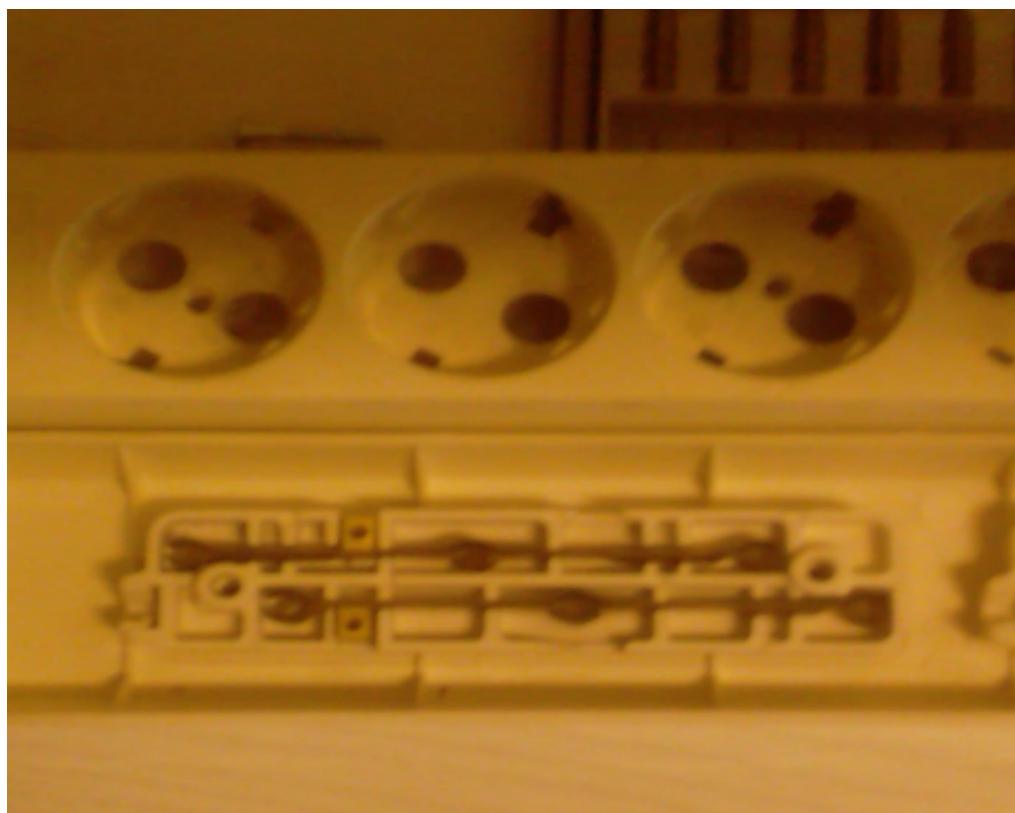
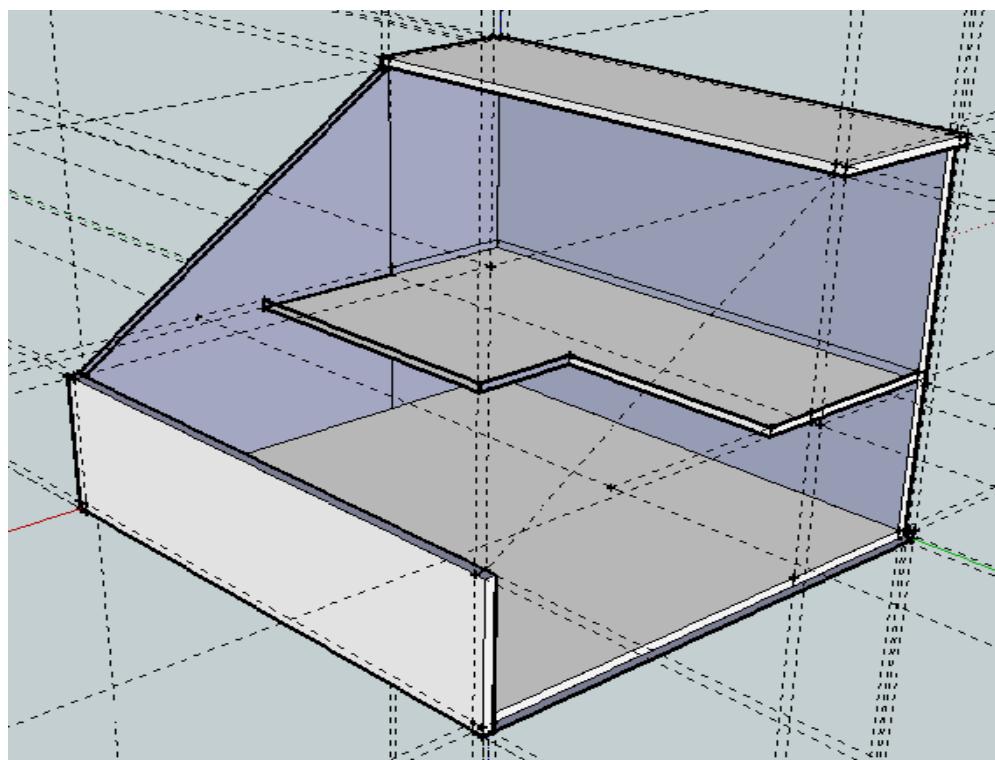


Figure 3.24 – Second version of jacks



*Figure 3.25 – Jacks after modified*

We used pleksiglass for constructing new box. First we design 3D box with google Sketchup. Because of milimeters are important while integrating parts of box and we get more professional system box. Our system box cut a fine figure.



*Figure 3.26 – 3D model of system box*

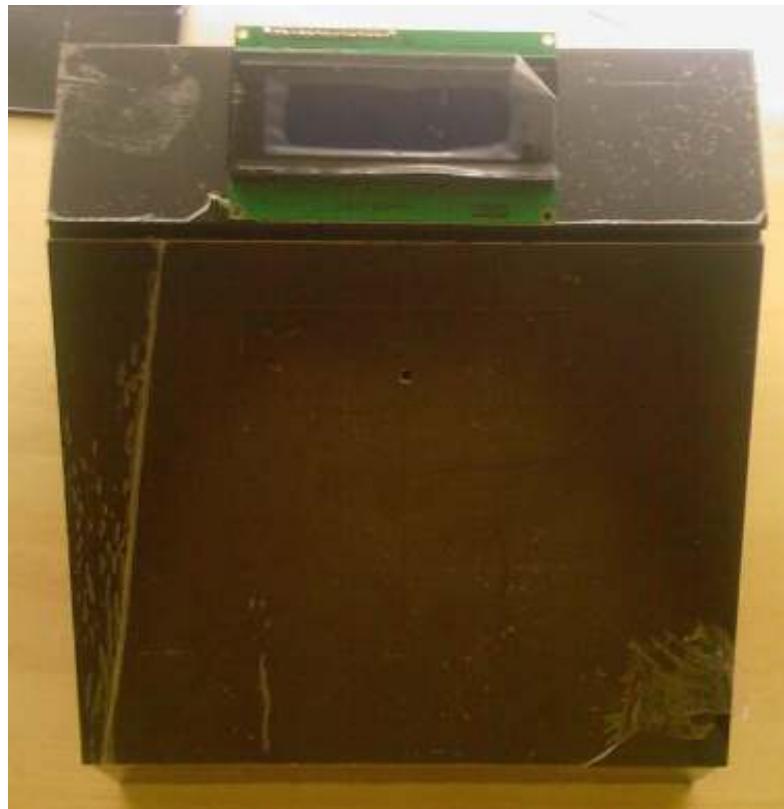


Figure 3.27 – New system box

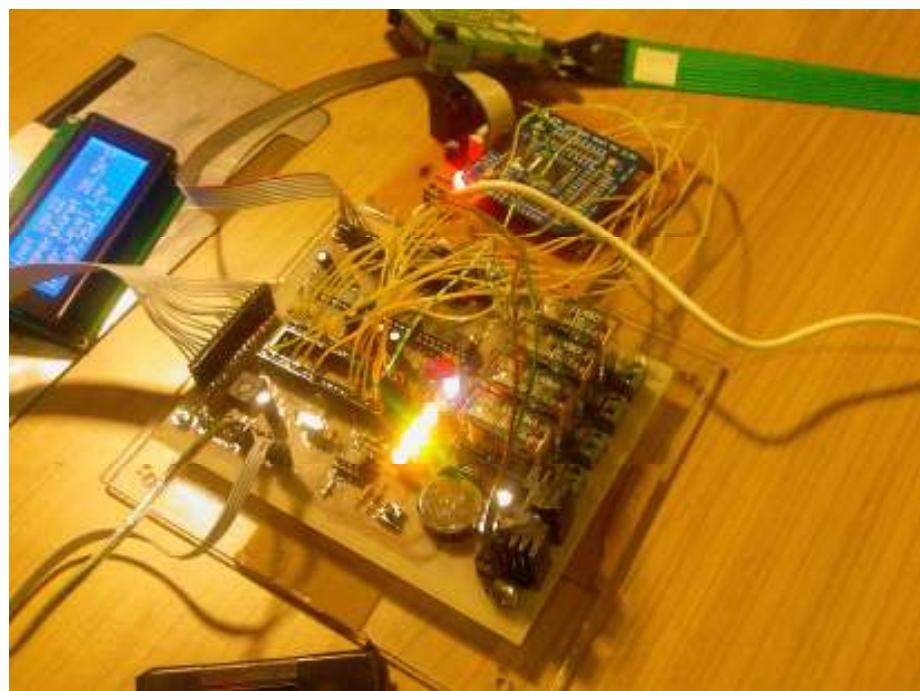


Figure 3.28 – SurfingFish board

## 3.2 Web

### 3.2.1 Architectural Overview

The Microsoft.NET Framework is a multi language platform for building, deploying, and running Web Services and applications. ASP:NET is a compiled, .NET – based environment, allowing authors to write applications in any .NET compatible language, including Visual Basic.NET and Jscript.NET. For this project, C# was the language used.

#### 3.2.1.1 Technology Overview:ASP.NET Web Application

##### Pages

.NET pages, known officially as "web forms", are the main building block for application development. Web forms are contained in files with an ".aspx" extension; in programming jargon, these files typically contain static (X)HTML markup, as well as markup defining server-side Web Controls and User Controls where the developers place all the required static and dynamic content for the web page. Additionally, dynamic code which runs on the server can be placed in a page within a block <% -- dynamic code -- %> which is similar to other web development technologies such as PHP, JSP, and ASP, but this practice is generally discouraged except for the purposes of data binding since it requires more calls when rendering the page.

##### Code-behind model

It is recommended by Microsoft for dealing with dynamic program code to use the code-behind model, which places this code in a separate file or in a specially designated script tag. Code-behind files typically have names like *MyPage.aspx.cs* or *MyPage.aspx.vb* based on the ASPX file name (this practice is automatic in Microsoft Visual Studio and other IDEs). When using this style of programming, the developer writes code to respond to different events, like the page being loaded, or a control being clicked, rather than a procedural walk through the document.

ASP.NET's code-behind model marks a departure from Classic ASP in that it encourages developers to build applications with separation of presentation and content in mind. In theory, this would allow a web designer, for example, to focus on the design

markup with less potential for disturbing the programming code that drives it. This is similar to the separation of the controller from the view in model-view-controller frameworks.

### User controls

ASP.NET supports creating reusable components through the creation of User Controls. A User Control follows the same structure as a Web Form, except that such controls are derived from the System.Web.UI.UserControl class, and are stored in ASCX files. Like ASPX files, an ASCX file contains static HTML or XHTML markup, as well as markup defining web control and other User Controls. The code-behind model can be used.

Programmers can add their own properties, methods,<sup>[9]</sup> and event handlers.<sup>[10]</sup> An event bubbling mechanism provides the ability to pass an event fired by a user control up to its containing page.

User can also build Custom Controls for Asp.Net application. Where controls are in compiled DLL file. And by using Register directive user can use control from DLL.

### Rendering technique

ASP.NET uses a *visited composites* rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template. Literal text goes into instances of the Literal control class, and server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page. The page doubles as the root of the control tree.

Actual requests for the page are processed through a number of steps. First, during the initialization steps, an instance of the page class is created and the initialization code is executed. This produces the initial control tree which is now typically manipulated by the methods of the page in the following steps. As each node in the tree is a control represented as an instance of a class, the code may change the tree structure as well as

manipulate the properties/methods of the individual nodes. Finally, during the rendering step a visitor is used to visit every node in the tree, asking each node to render itself using the methods of the visitor. The resulting HTML output is sent to the client.

After the request has been processed, the instance of the page class is discarded and with it the entire control tree.

### **State management**

ASP.NET applications are hosted in a web server and are accessed over the stateless HTTP protocol. As such, if the application uses stateful interaction, it has to implement state management on its own. ASP.NET provides various functionality for state management in ASP.NET applications.

### **Application state**

Application state is a collection of user-defined variables that are shared by an ASP.NET application. These are set and initialized when the Application\_OnStart event fires on the loading of the first instance of the applications and are available till the last instance exits. Application state variables are accessed using the Applications collection, which provides a wrapper for the application state variables. Application state variables are identified by names.

### **Session state**

Session state is a collection of user-defined session variables, which are persisted during a user session. These variables are unique to different instances of a user session, and are accessed using the Session collection. Session variables can be set to be automatically destroyed after a defined time of inactivity, even if the session does not end. At the client end, a user session is identified either by a cookie or by encoding the session ID in the URL itself. ASP.NET supports three modes of persistence for session variables

## In Process Mode

The session variables are maintained within the ASP.NET process. This is the fastest way; however, in this mode the variables are destroyed when the ASP.NET process is recycled or shut down. Since the application is recycled from time to time this mode is not recommended for critical applications.

## ASPState Mode

In this mode, ASP.NET runs a separate Windows service that maintains the state variables. Because the state management happens outside the ASP.NET process and .NET Remoting must be utilized by the ASP.NET engine to access the data, this mode has a negative impact on performance in comparison to the In Process mode, although this mode allows an ASP.NET application to be load-balanced and scaled across multiple servers. However, since the state management service runs independent of ASP.NET, the session variables can persist across ASP.NET process shutdowns.

## SqlServer Mode

In this mode, the state variables are stored in a database server, accessible using SQL. Session variables can be persisted across ASP.NET process shutdowns in this mode as well. The main advantage of this mode is it would allow the application to balance load on a server cluster while sharing sessions between servers.

## View state

View state refers to the page-level state management mechanism, which is utilized by the HTML pages emitted by ASP.NET applications to maintain the state of the web form controls and widgets. The state of the controls are encoded and sent to the server at every form submission in a hidden field known as `__VIEWSTATE`. The server sends back the variable so that when the page is re-rendered, the controls render at their last state. At the server side, the application might change the viewstate, if the processing results in

updating the state of any control. The states of individual controls are decoded at the server, and are available for use in ASP.NET pages using the ViewState collection.

## Other

Other means of state management that are supported by ASP.NET are cookies, caching, and using the query string.

## Template engine

When first released, ASP.NET lacked a template engine. Because the .NET framework is object-oriented and allows for inheritance, many developers would define a new base class that inherits from "System.Web.UI.Page", write methods here that render HTML, and then make the pages in their application inherit from this new class. While this allows for common elements to be reused across a site, it adds complexity and mixes source code with markup. Furthermore, this method can only be visually tested by running the application - not while designing it. Other developers have used include files and other tricks to avoid having to implement the same navigation and other elements in every page. ASP.NET 2.0 introduced the concept of "master pages", which allow for template-based page development. A web application can have one or more master pages, which beginning with ASP.NET 3.5, can be nested.<sup>[14]</sup> Master templates have place-holder controls, called *ContentPlaceHolders* to denote where the dynamic content goes, as well as HTML and JavaScript shared across child pages.

Child pages use those ContentPlaceHolder controls, which must be mapped to the place-holder of the master page that the content page is populating. The rest of the page is defined by the shared parts of the master page, much like a mail merge in a word processor. All markup and server controls in the content page must be placed within the ContentPlaceHolder control.

When a request is made for a content page, ASP.NET merges the output of the content page with the output of the master page, and sends the output to the user.

The master page remains fully accessible to the content page. This means that the content page may still manipulate headers, change title, configure caching etc. If the master page exposes public properties or methods (e.g. for setting copyright notices) the content page can use these as well. [5]

### **3.2.1.2 Technology Overview : AJAX**

Ajax (also known as AJAX), shorthand for "Asynchronous JavaScript and XML," is a web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is intended to increase the web page's interactivity, speed, and usability. The first use of the term in public was by Jesse James Garrett in February 2005. Garrett thought of the term when he realized the need for a shorthand term to represent the suite of technologies he was proposing to a client. Although the term Ajax was coined in 2005, most of the technologies that enable Ajax started a decade earlier with Microsoft's initiatives in developing Remote Scripting. Techniques for the asynchronous loading of content on an existing Web page without requiring a full reload date back as far as the IFRAME element type (introduced in Internet Explorer 3 in 1996) and the LAYER element type (introduced in Netscape 4 in 1997, abandoned during early development of Mozilla). Both element types had a src attribute that could take any external URL, and by loading a page containing JavaScript that manipulated the parent page, Ajax-like effects could be attained. This set of client-side technologies was usually grouped together under the generic term of DHTML. Macromedia's Flash could also, from version 4, load XML and CSV files from a remote server without requiring a browser being refreshed.

#### **Advantages**

#### **User interface**

The most obvious reason for using Ajax is an improvement to the user experience. Pages using Ajax behave more like a standalone application than a typical web page. Clicking on links that cause the entire page to refresh feels like a "heavy" operation. With Ajax, the page often can be updated dynamically, allowing a faster response to the user's interaction. While the full potential of Ajax has yet to be determined, some believe it will

prove to be an important technology, helping make the Web even more interactive and popular than it currently is. Bandwidth usage By generating the HTML locally within the browser and only bringing down JavaScript calls and the actual data, Ajax web pages can appear to load relatively quickly since the payload coming down is much smaller in size. An example of this technique is a large result set where multiple pages of data exist. With Ajax, the HTML of the page (e.g., a table structure with related TD and TR tags) can be produced locally in the browser and not brought down with the first page of the document. In addition to "load on demand" of contents, some web-based applications load stubs of event handlers and then load the functions on the fly. This technique significantly cuts down the bandwidth consumption for web applications that have complex logic and functionality. [6]

Separation of data, format, style, and function a less specific benefit of the Ajax approach is that it tends to encourage programmers to clearly separate the methods and formats used for the different aspects of information delivery via the web. Although Ajax can appear to be a jumble of languages and techniques, and programmers are free to adopt and adapt whatever works for them, they are generally propelled by the development motive itself to adopt separation between the following:

- \* Adopt separation between the raw data or content to be delivered - which is normally embedded in XML and sometimes derived from a server-side database.
- \* Adopt separation between the format or structure of the webpage - which is almost always built in HTML (or better, XHTML) and is then reflected and made available to dynamic manipulation in the DOM.
- \* Adopt separation between the style elements of the webpage: everything from fonts to picture placement are derived by reference to embedded or referenced CSS.
- \* Adopt separation between the functionality of the web page which is provided by a combination of
  1. Javascript on the client browser (also called DHTML),
  2. Standard HTTP and XMLHttpRequest for client-to-server communication, and

3. Server-side scripting and/or programs using any suitable language preferred by the programmer to receive the client's specific requests and respond appropriately.

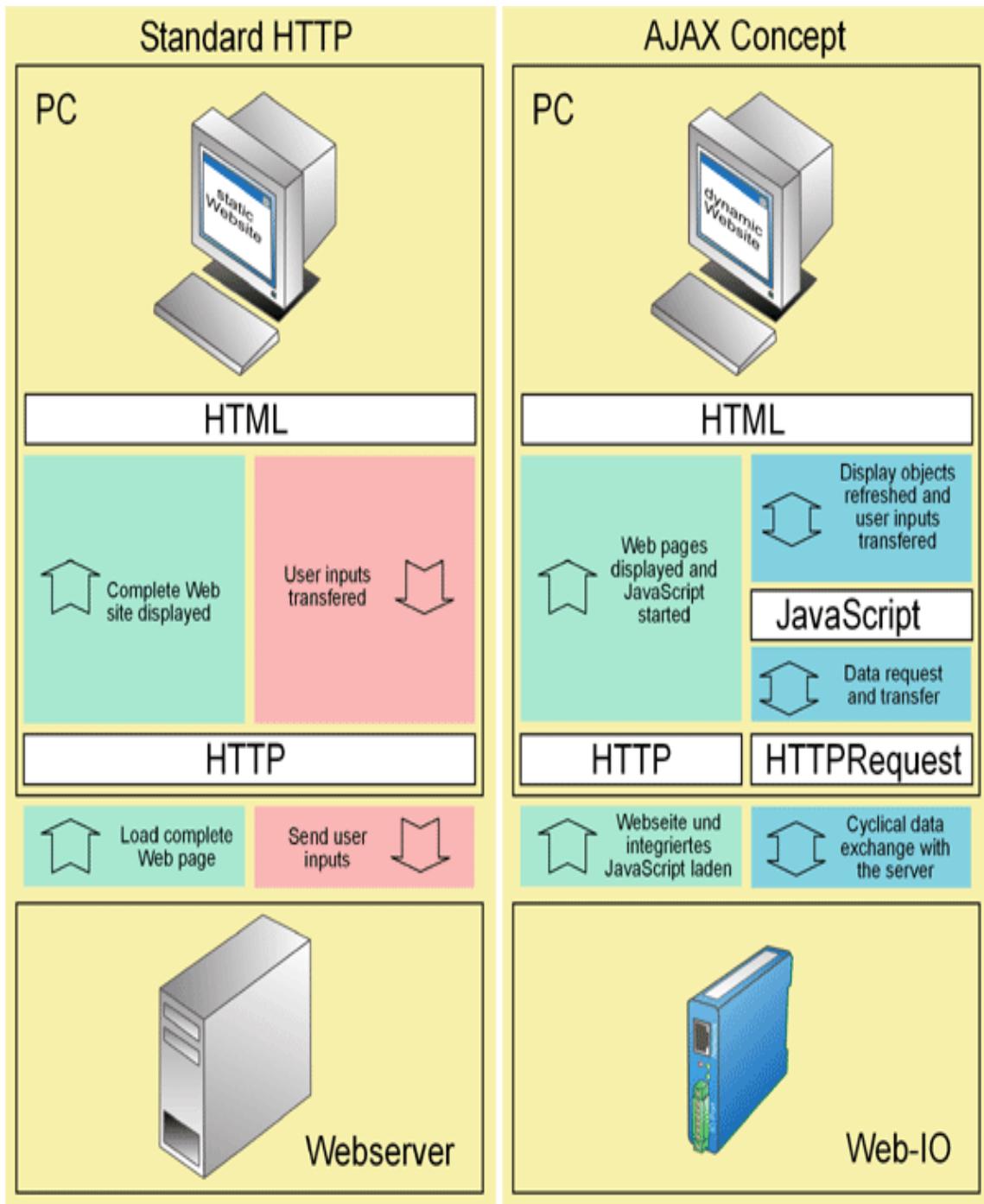


Figure 3.29 – Ajax and http diagram

### 3.2.1.3 Technology Overview : MySQL(Database)

.MySQL was designed to work with small and mid-sized databases. We believe MySQL is better than other Web database options because along with being the most widely used and best supported Web database, it is a true relational database. And in this Project we used mysql database[7]



Figure 3.30 – MySQL logo

### 3.2.1.4 Software Overview: "Whidbey" Visual Studio.NET (IDE)

We used visual studio ide for this project because of several reasons;

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring.. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It allows plug-ins to be added that enhance the functionality at almost every level [8]

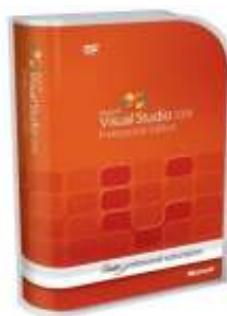
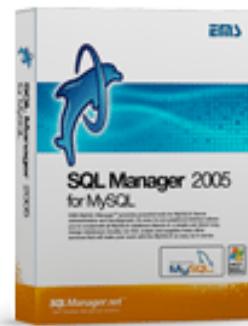


Figure 3.31 –Visual Studio

### 3.2.1.5 Software Overview: EMS SQL Manager For MySQL

We used EMS SQL Manager for MySQL for this project. It could be a good choice for us for a number of reasons:

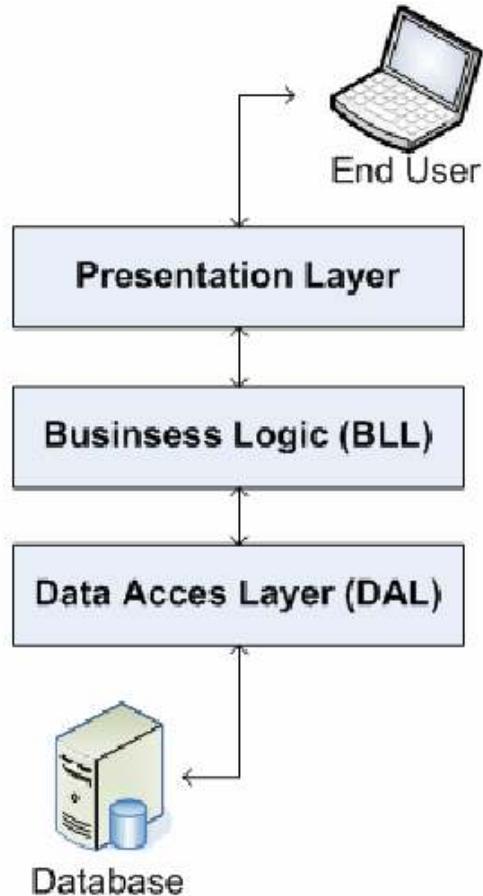


*Figure 3.32 – Sql Manager for mysql*

#### Key Features

- Full support of MySQL versions from 3.23 to 6.0
- Support of UTF8 data
- Rapid database management and navigation
- Simple management of all MySQL objects
- Advanced data manipulation tools
- Powerful security management
- Excellent visual and text tools for query building
- Impressive data export and import capabilities
- Report designer with clear in use report construction wizard
- Powerful Visual Database Designer
- Easy-to-use wizards performing MySQL services [9]

### 3.2.2 The 3-Tier approach



*Figure 3.33 – 3-Tier approach*

The purpose of the 3 tier approach is to divide the design into layers that have different purposes. Each layer can then be independently worked on, improved, redesigned and so on. Only the interfaces between the layers need to stay as they are if we don't want change in one layer to affect the others.

Using this approach has several benefits:

- Scalability – Splitting the design into layers makes it much easier for the web project to grow bigger in size. At first all layers can be on one server. Later, if the website is successful and attracts many visitors, one layer can be extracted to another server to balance the load, and so on. Different programmers can work on different tiers –no programmer really has to know the project from end to end.

- Robustness – The project in a tiered approach is less sensitive to failure, and therefore more robust. When something goes wrong, the responsible layer is immediately recognized, and knowing what to fix is simpler and faster in comparison to a project that has all the logic in a single layer. Moreover, intrusion is also more difficult in a multi layered environment, because there are more layers for the intruder to penetrate. Getting all the way to the database is harder in a layered design.

### 3.2.3 The Database

We used Mysql as our database. Here is the database diagram

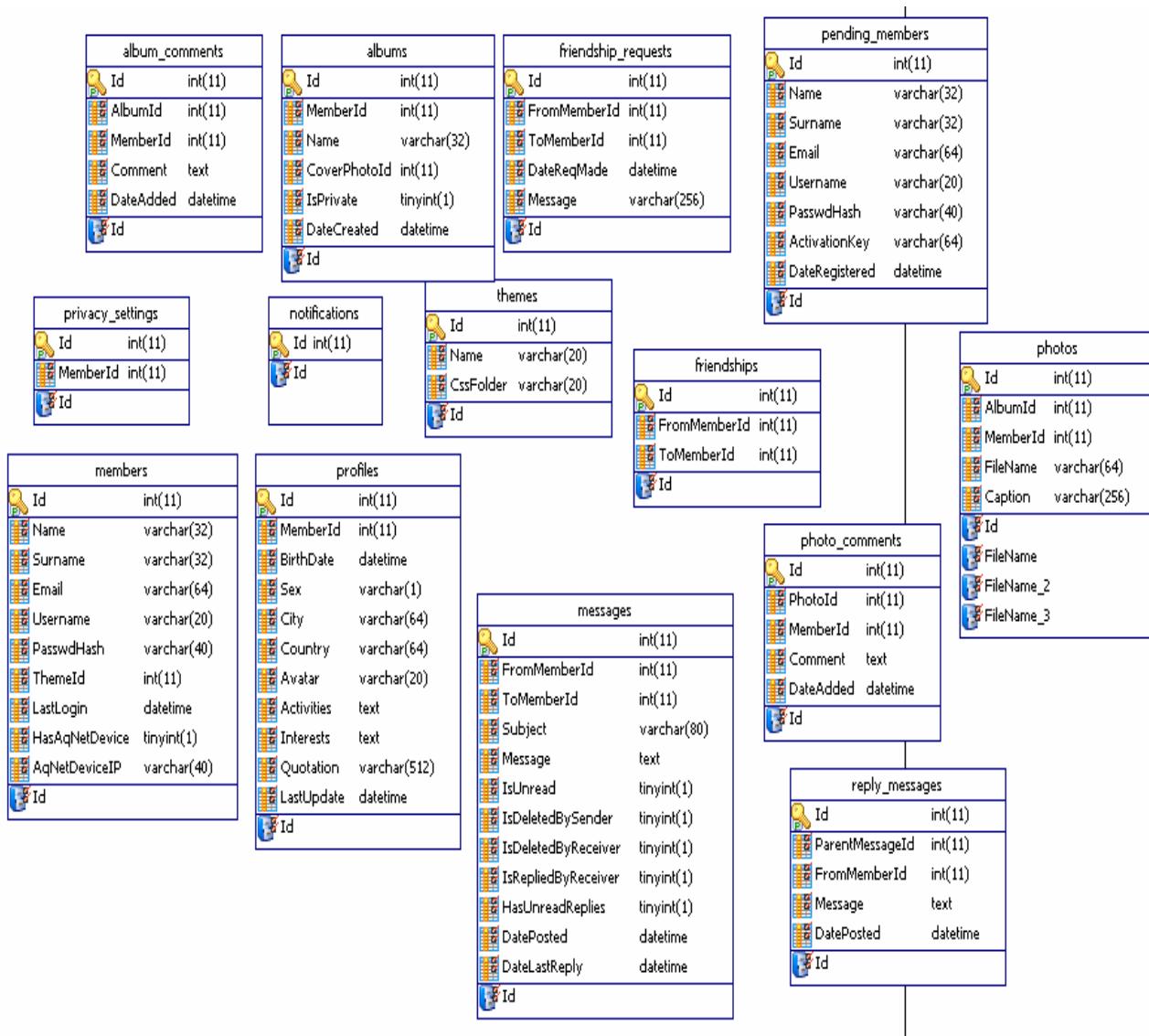


Figure 3.34 – Database diagram

### 3.2.3.1 Tables

Table - [members] - [aquanetdb on localhost]							
Properties	Fields	Indices	Foreign Keys	Triggers	Data	Dependencies	DDL
Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
<input checked="" type="checkbox"/> id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/> Name	VARCHAR	32	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	"	Member's name
<input checked="" type="checkbox"/> Surname	VARCHAR	32	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	"	Member's surname
<input checked="" type="checkbox"/> Email	VARCHAR	64	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	primary contact email address
<input checked="" type="checkbox"/> Username	VARCHAR	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	"	Member's username, used for login.
<input checked="" type="checkbox"/> PasswdHash	VARCHAR	40	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	"	Hash value of member's password
<input checked="" type="checkbox"/> Themeld	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		member's theme choice
<input checked="" type="checkbox"/> LastLogin	DATETIME	0	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	last time user was logged in
<input checked="" type="checkbox"/> HasAqNetDevice	TINYINT	1	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0	whether or not the member has an aquanet device
<input checked="" type="checkbox"/> AqNetDeviceIP	VARCHAR	40	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	aquanet device IP address if the device is online

Table 3.2 – Members table

**Table – members:** Upon registration, each user gets a unique user id (id) which acts as the primary key in the table. The table stores username and password for the log in the website.

Table - [friendship_requests] - [aquanetdb on localhost]							
Properties	Fields	Indices	Foreign Keys	Triggers	Data	Dependencies	DDL
Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
<input checked="" type="checkbox"/> id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		request id
<input checked="" type="checkbox"/> FromMemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		who makes the request
<input checked="" type="checkbox"/> ToMemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		to whom the request was made
<input checked="" type="checkbox"/> DateReqMade	DATETIME	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		when the friendship request was made
<input checked="" type="checkbox"/> Message	VARCHAR	256	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	an optional message while making the friendship request

Table 3.3 – Friendship\_request table

**Table – friendship\_request:** This table represents the friendship requests between the members of surfingfish site. Message field holds the optional message text while making the friendship request. DateReqMade holds an optional message while making the friendship request. FromMemberID and ToMemberId represents who makes the request and to whom the request was made and each request gets unique id(Id)

Table - [friendships] - [aquanetdb on localhost]							
Properties	Fields	Indices	Foreign Keys	Triggers	Data	Dependencies	DDL
Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
<input checked="" type="checkbox"/> id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		friendship id
<input checked="" type="checkbox"/> FromMemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		memberid of the member who made the friendship request
<input checked="" type="checkbox"/> ToMemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		memberid of the member to whom the friendship request was made

Table 3.4 – Friendship table

**Table - friendship:** This table defines a friendship relationship between pairs of member. FromMemberID represents the member who made the friendship request and ToMemeberID represents whom the friendship request was made and each friendship get unique id(Id)

The screenshot shows the 'Properties' tab of the MySQL Workbench interface for the 'messages' table. The table has 11 columns:

Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
Iid	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		message id
FromMemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		who sent the message
ToMemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		who received the message
Subject	VARCHAR	80	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	subject of the message
Message	TEXT	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		the message
IsUnread	TINYINT	1	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		whether or not the message or any reply of it is ready by the receiver
IsDeletedBySender	TINYINT	1	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		whether or not the sender deleted the message
IsDeletedByReceiver	TINYINT	1	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		whether or not the message is deleted by the receiver
IsRepliedByReceiver	TINYINT	1	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		whether or not the receiver sent a reply to this message
HasUnreadReplies	TINYINT	1	0	<input type="checkbox"/>	<input type="checkbox"/>	0	
DatePosted	DATETIME	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		when the message was posted
DateLastReply	DATETIME	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		when last reply is sent to this message, if no replies exist this field is equal to DatePosted

Table 3.5 – Messages table

**Table – messages:** This table is in charge of holding private messages. Each new message gets unique identifier Iid) which is the primary key of the table. ToMemberId represents the person who got the message, and FromMemberId represents the person who sent the message. This table also stores message's post date, and who deleted message data.

The screenshot shows the 'Properties' tab of the MySQL Workbench interface for the 'reply\_messages' table. The table has 5 columns:

Field Name	Field Type	Size	Prec...	Not Null	Unsigned	Default	Comment
Iid	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		reply message id
ParentMessageId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		to which message this reply message is sent
FromMemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		who sent the message
Message	TEXT	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		message content
DatePosted	DATETIME	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		when this reply is posted

Table 3.6 – Reply\_messages table

**Table – reply\_messages:** This table is holding reply messages, Message field reperesents the message content, ParentMessageId represents to which message this reply message is sent, FromMemeberId field represnts to who sent the message and DatePosted field holds the posted time of the reply message.

The screenshot shows the AquanetDB interface with the title "Table - [albums] - [aquanetdb on localhost]". Below the title is a navigation bar with tabs: Properties, Fields, Indices, Foreign Keys, Triggers, Data, Dependencies, and DDL. The "Fields" tab is selected. The main area is a table with the following columns: Field Name, Field Type, Size, Precision, Not Null, Unsigned, Default, and Comment. The table contains six rows of data:

Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
Id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		album id
MemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		who created the album
Name	VARCHAR	32	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	"	album's name
CoverPhotoId	INTEGER	11	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Null	album cover photo id
IsPrivate	TINYINT	1	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	whether or not the album is private
DateCreated	DATETIME	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		when the album was created

Table 3.7 –Albums table

**Table – albums:** This table holds information about the albums that have been added.

Every album gets unique identifier (Id), and the user who uploaded is represented by MemberID. DateCreated field is represent the create date of the album and IsPrivate field is represent album's privacy (only friends can see the albums or everyone can see the albums).

The screenshot shows the AquanetDB interface with the title "Table - [album\_comments] - [aquanetdb on localhost]". Below the title is a navigation bar with tabs: Properties, Fields, Indices, Foreign Keys, Triggers, Data, Dependencies, and DDL. The "Fields" tab is selected. The main area is a table with the following columns: Field Name, Field Type, Size, Precision, Not Null, Unsigned, Default, and Comment. The table contains five rows of data:

Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
Id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		comment id
AlbumId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		album id
MemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		member who made the comment
Comment	TEXT	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		album comment text
DateAdded	DATETIME	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		when the comment is added

Table 3.8 – Album\_comments table

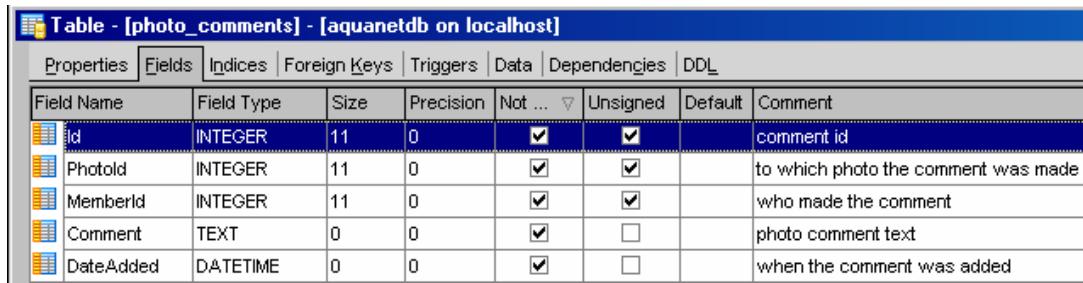
**Table – album\_comments:** This table holds comments of the albums. MemberId represents the member who made the comment, and Comment represents the album comment text and DateAdded represents when the comment is added. AlbumId represents which album the comment was made.

The screenshot shows the AquanetDB interface with the title "Table - [photos] - [aquanetdb on localhost]". Below the title is a navigation bar with tabs: Properties, Fields, Indices, Foreign Keys, Triggers, Data, Dependencies, and DDL. The "Fields" tab is selected. The main area is a table with the following columns: Field Name, Field Type, Size, Precision, Not Null, Unsigned, Default, and Comment. The table contains five rows of data:

Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
Id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		photo id
AlbumId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		which album this photo belongs to
MemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		who added the photo
FileName	VARCHAR	64	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	"	photo file name
Caption	VARCHAR	256	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	caption for the photo

Table 3.9 – Photos table

**Table – photos:** This table hold the information about photos. Every photo gets unique identifier (Id).This table stores; photo file name (FileName), who added photo (MemberId), which album this photo belongs to (AlbumId) and caption for the photo (Caption).

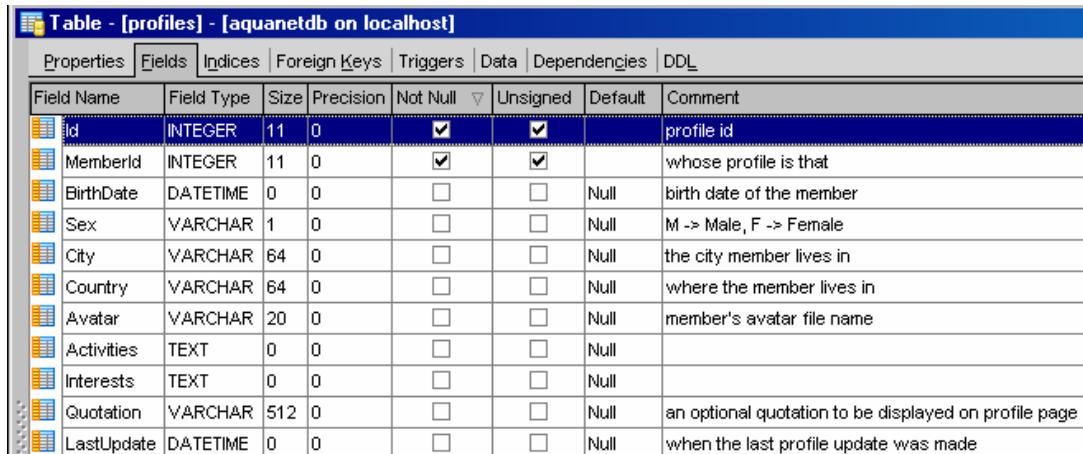


The screenshot shows the AquanetDB interface with the title 'Table - [photo\_comments] - [aquanetdb on localhost]'. Below the title is a navigation bar with tabs: Properties, Fields (which is selected), Indices, Foreign Keys, Triggers, Data, Dependencies, and DDL. The main area is a table with the following columns: Field Name, Field Type, Size, Precision, Not Null, Unsigned, Default, and Comment. The table contains five rows of data:

Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
Id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		comment id
PhotoId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		to which photo the comment was made
MemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		who made the comment
Comment	TEXT	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		photo comment text
DateAdded	DATETIME	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		when the comment was added

Table 3.10 – Photo\_comments table

**Table – photo\_comments:** This table holds comments of the photos. MemberId represents the member who made the comment, and Comment represents the album comment text and DateAdded represents when the comment is added. PhotoId represents which photo the comment was made.



The screenshot shows the AquanetDB interface with the title 'Table - [profiles] - [aquanetdb on localhost]'. Below the title is a navigation bar with tabs: Properties, Fields (selected), Indices, Foreign Keys, Triggers, Data, Dependencies, and DDL. The main area is a table with the following columns: Field Name, Field Type, Size, Precision, Not Null, Unsigned, Default, and Comment. The table contains ten rows of data:

Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment
Id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		profile id
MemberId	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		whose profile is that
BirthDate	DATETIME	0	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	birth date of the member
Sex	VARCHAR	1	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	M -> Male, F -> Female
City	VARCHAR	64	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	the city member lives in
Country	VARCHAR	64	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	where the member lives in
Avatar	VARCHAR	20	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	member's avatar file name
Activities	TEXT	0	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	
Interests	TEXT	0	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	
Quotation	VARCHAR	512	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	an optional quotation to be displayed on profile page
LastUpdate	DATETIME	0	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	when the last profile update was made

Table 3.11 – Profiles table

**Table – profiles:** This table holds personal information of the member like birth date, sex, country, interest etc.

Table - [themes] - [aquanetdb on localhost]								
Properties	Fields	Indices	Foreign Keys	Triggers	Data	Dependencies	DDL	
Field Name	Field Type	Size	Precision	Not Null	Unsigned	Default	Comment	
<input checked="" type="checkbox"/> Id	INTEGER	11	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		theme id	
<input checked="" type="checkbox"/> Name	VARCHAR	20	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	theme name	
<input checked="" type="checkbox"/> CssFolder	VARCHAR	20	0	<input type="checkbox"/>	<input type="checkbox"/>	Null	comma separated css file names for this theme	

Table 3.12 –Themes table

**Table – themes:** This table holds the theme information. Name field represents the theme name and CssFolder represents css file names for this theme.

### 3.2.3.2 Stored Procedures

A stored procedure is a subroutine available to applications accessing a relational database system. Stored procedures (sometimes called a proc, sproc, StoPro, or SP) are actually stored in the database data dictionary.

Typical uses for stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures are used to consolidate and centralize logic that was originally implemented in applications. Large or complex processing that might require the execution of several SQL statements is moved into stored procedures and all applications call the procedures only.

We used stored procedures as the database interface for the DAL. Here is the stored procedures diagram.

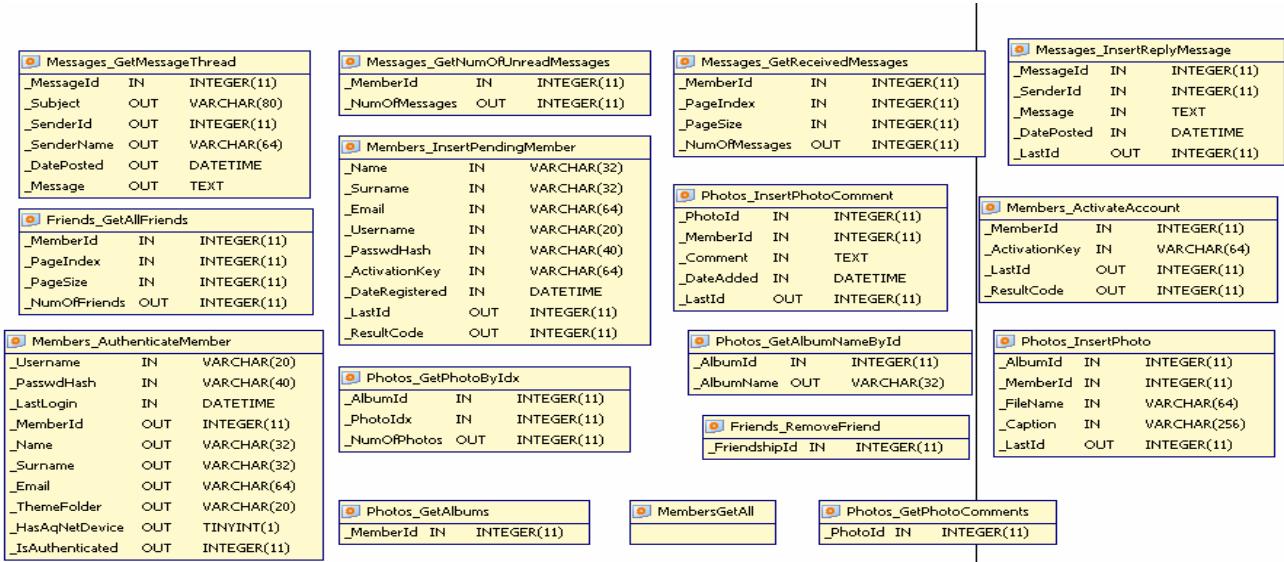


Figure 3.35 – Stored procedure diagram

### **Procedure - Friends\_RemoveFriends:**

<b>Parameters</b>		
Name	Type	Input/Output
_FriendshipId	INTEGER(11)	IN

<b>Description</b>
--------------------

*Table 3.13 – Friends stored procedure table*

This procedure gets \_FriendshipId as an input parameter and deletes the friend from the friendship table.

### **Procedure - MembersGetAll:**

<b>Description</b>
--------------------

This procedure hasn't got any parameters. This procedure gets all members from the members table and orders the members by member id.

*Table 3.14 – Memebers get all stored procedure table*

### **Procedure – Members\_ActivateAccount:**

<b>Parameters</b>		
Name	Type	Input/Output
_MemberId	INTEGER(11)	IN
_ActivationKey	VARCHAR(64)	IN
_LastId	INTEGER(11)	OUT
_ResultCode	INTEGER(11)	OUT

<b>Description</b>
--------------------

*Table 3.15 – Activate account stored procedure table*

This procedure activates the new member of the site with the specified member id (\_MemberId) and activation key (\_ActivationKey) and returns result of the activation (\_ResultCode) and \_LastId through the output parameters.

### **Procedure – Members\_AuthenticateMember:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_Username	VARCHAR(20)	IN
_PasswdHash	VARCHAR(40)	IN
_LastLogin	DATETIME	IN
_MemberId	INTEGER(11)	OUT
_Name	VARCHAR(32)	OUT
_Surname	VARCHAR(32)	OUT
_Email	VARCHAR(64)	OUT
_ThemeFolder	VARCHAR(20)	OUT
_HasAqNetDevice	TINYINT(1)	OUT
_IsAuthenticated	INTEGER(11)	OUT

<b>Description</b>
--------------------

*Table 3.16 –Authenticate member stored procedure table*

This procedure authenticates the member and returns with member's private information.

### **Procedure – Members\_InsertPendingMembers:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_Name	VARCHAR(32)	IN
_Surname	VARCHAR(32)	IN
_Email	VARCHAR(64)	IN
_Username	VARCHAR(20)	IN
_PasswdHash	VARCHAR(40)	IN
_ActivationKey	VARCHAR(64)	IN
_DateRegistered	DATETIME	IN
_LastId	INTEGER(11)	OUT
_ResultCode	INTEGER(11)	OUT

<b>Description</b>
--------------------

*Table 3.17 – Insert pending stored procedure table*

This procedure inserts a new row in the pending\_member table and returns the LastId and \_ResultCode of the added row through the output parameter and It takes input values for all fields of this table.

### **Procedure – Messages\_GetMessageThreads:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_MessageId	INTEGER(11)	IN
_Subject	VARCHAR(80)	OUT
_SenderId	INTEGER(11)	OUT
_SenderName	VARCHAR(64)	OUT
_DatePosted	DATETIME	OUT
_Message	TEXT	OUT

<b>Description</b>		
--------------------	--	--

*Table 3.18 –Get message stored procedure table*

This procedure gets \_MessageId as an input parameter and returns with some information of the message (posted date (\_DatePosted), name of the sender (\_SenderId), Subject of the message (\_Subject), Message (\_Message), sender's Id (\_SenderId) )

### **Procedure – Messages\_GetNumberOfUnreadMessages:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_MemberId	INTEGER(11)	IN
_NumOfMessages	INTEGER(11)	OUT

<b>Description</b>		
--------------------	--	--

*Table 3.19 – Unread messages stored procedure table*

This procedure gets \_MemberId as an input parameter and returns value of the number of messages (\_NumofMessages) through the output parameter.

### **Procedure – Messages\_GetReceivedMessages:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_MemberId	INTEGER(11)	IN
_PageIndex	INTEGER(11)	IN
_PageSize	INTEGER(11)	IN
_NumOfMessages	INTEGER(11)	OUT

<b>Description</b>		
--------------------	--	--

*Table 3.20 –Recieved messages stored procedure table*

This procedure gets \_MemberId and page attributes (\_PageIndex, \_PageSize ) and returns value of the number of messages(\_NumOfMessages) through the output parameter

### Procedure – Messages\_InsertReplyMessage:

Parameters		
Name	Type	Input/Output
_MessageId	INTEGER(11)	IN
_SenderId	INTEGER(11)	IN
_Message	TEXT	IN
_DatePosted	DATETIME	IN
_LastId	INTEGER(11)	OUT

Description
-------------

*Table 3.21 – Reply messages stored procedure table*

This procedure inserts a new row in the reply\_messages table and returns the LastId of the added row through the output parameter and It gets send date of the message (\_DatePosted), sender's id(\_SenderId), message id (\_MessageId) and message(\_Message)

### Procedure – Photos\_GetAlbumNameById:

Parameters		
Name	Type	Input/Output
_AlbumId	INTEGER(11)	IN
_AlbumName	VARCHAR(32)	OUT

Description
-------------

*Table 3.22 – Album name stored procedure table*

This procedure gets album name by album's id. The procedure gets an album id (\_AlbumID) as an input parameter and returns the name of the album (\_AlbumName) through output parameter.

### Procedure – Photos\_GetAlbums:

Parameters		
Name	Type	Input/Output
_MemberId	INTEGER(11)	IN

Description
-------------

*Table 3.23 – Get albums stored procedure table*

This procedure gets albums from the albums table by member id (\_MemberID).

### **Procedure – Photos\_GetPhotoByIdx:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_AlbumId	INTEGER(11)	IN
_PhotoIdx	INTEGER(11)	IN
_NumOfPhotos	INTEGER(11)	OUT

<b>Description</b>
--------------------

*Table 3.24 – Get photo stored procedure table*

This procedure returns number of photos (\_NumofPhotos) whose album Id (AlbumId) and photo index (\_PhotoIdx) is passed as an input.

### **Procedure – Photos\_GetPhotoComments:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_PhotoId	INTEGER(11)	IN

<b>Description</b>
--------------------

*Table 3.25 – Photo comments stored procedure table*

This procedure gets the comments of photo by photo id (\_PhotoId) from the photo\_comments table.

### **Procedure – Photos\_InsertPhoto:**

<b>Parameters</b>		
<b>Name</b>	<b>Type</b>	<b>Input/Output</b>
_AlbumId	INTEGER(11)	IN
_MemberId	INTEGER(11)	IN
_FileName	VARCHAR(64)	IN
_Caption	VARCHAR(256)	IN
_LastId	INTEGER(11)	OUT

<b>Description</b>
--------------------

*Table 3.26 – Insert photo stored procedure table*

This procedure inserts a new row in the photos table and returns the LastId of the added row through the output parameter.

### Procedure - Photos\_InsertPhotoComment :

Parameters		
Name	Type	Input/Output
_PhotoId	INTEGER(11)	IN
_MemberId	INTEGER(11)	IN
_Comment	TEXT	IN
_DateAdded	DATETIME	IN
_LastId	INTEGER(11)	OUT

Description
-------------

Table 3.27 –Insert photo comment stored procedure table

This procedure inserts a new row in the photo\_comments table and returns the LastId of the added row through the output parameter.

### 3.2.4 DAL – Data Access Layer

#### 3.2.4.1 DAL introduction

The DAL acts as an abstraction layer between the business logic and the database. Our focus was on designing it so that adding functionality to it will be easy and simple. As web sites grow, it is only a matter of time until more functionality is needed, and when new tables are added we don't want them to be difficult to implement in the DAL.

#### 3.2.4.2 DAL classes

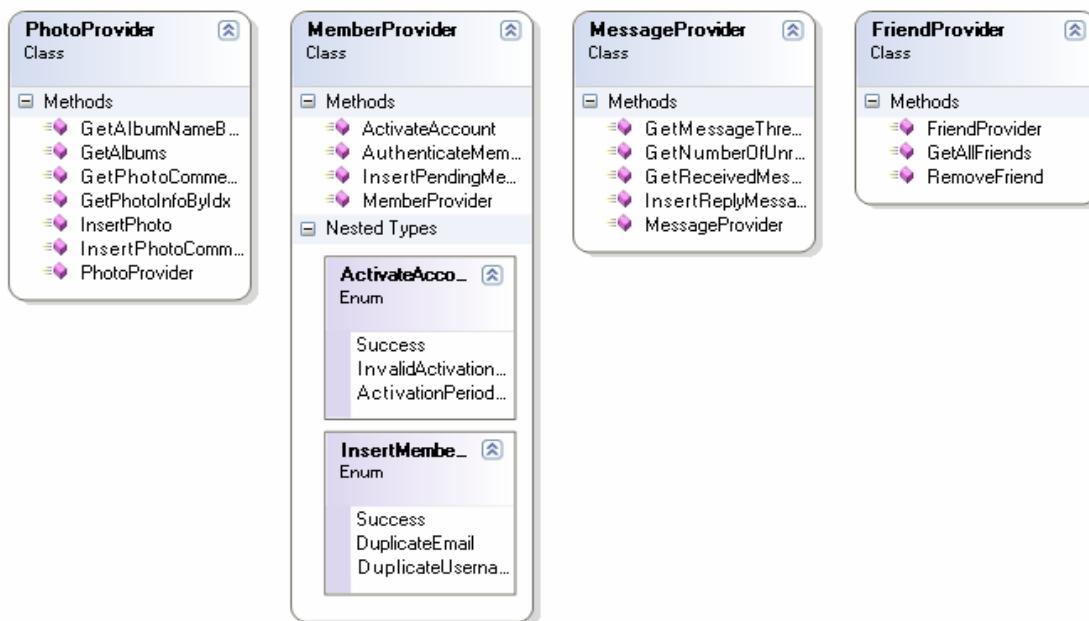


Figure 3.36 – DAL classes diagram

**PhotoProvider.cs:**

“PhotoProvider.cs” module provides access to database tables related to photo albums functionality. It contains public static functions and each one encapsulates a stored procedure. By calling those methods the business logic layer and the presentation layer can access and modify photo albums, photo comments related database tables.

**MemberProvider.cs:**

“MemberProvider.cs” module provides access to database tables related to website members. It contains public static functions and each one encapsulates a stored procedure. By calling those methods the business logic layer and the presentation layer can access and modify member registration, authentication etc. related database tables.

**MessageProvider.cs:**

“MessageProvider.cs” module provides access to database tables related to messages. It contains public static functions and each one encapsulates a stored procedure. By calling those methods the business logic layer and the presentation layer can access and modify message sending, message retrieval, message removal etc. related database tables.

**FriendProvider.cs:**

“FriendProvider.cs” module provides access to database tables related to friendship. It contains public static functions and each one encapsulates a stored procedure. By calling those methods the business logic layer and the presentation layer can access and modify friendship related database tables.

### **3.2.5 The Business Logic Layer**

The business classes are created directly under the ~/App\_Code folder, in a BLL subfolder so that they are automatically compiled at runtime, just like the pages. Business classes use the DAL classes to provide access to data and are mostly used to enforce

validation rules, check constraints, and provide an object-oriented representation of the data and methods to work with it. Thus, the BLL serves as a mapping layer that makes the underlying relational database appear as objects to user interface code. Relational databases are inherently not object oriented, so this BLL provides a far more useful representation of data.

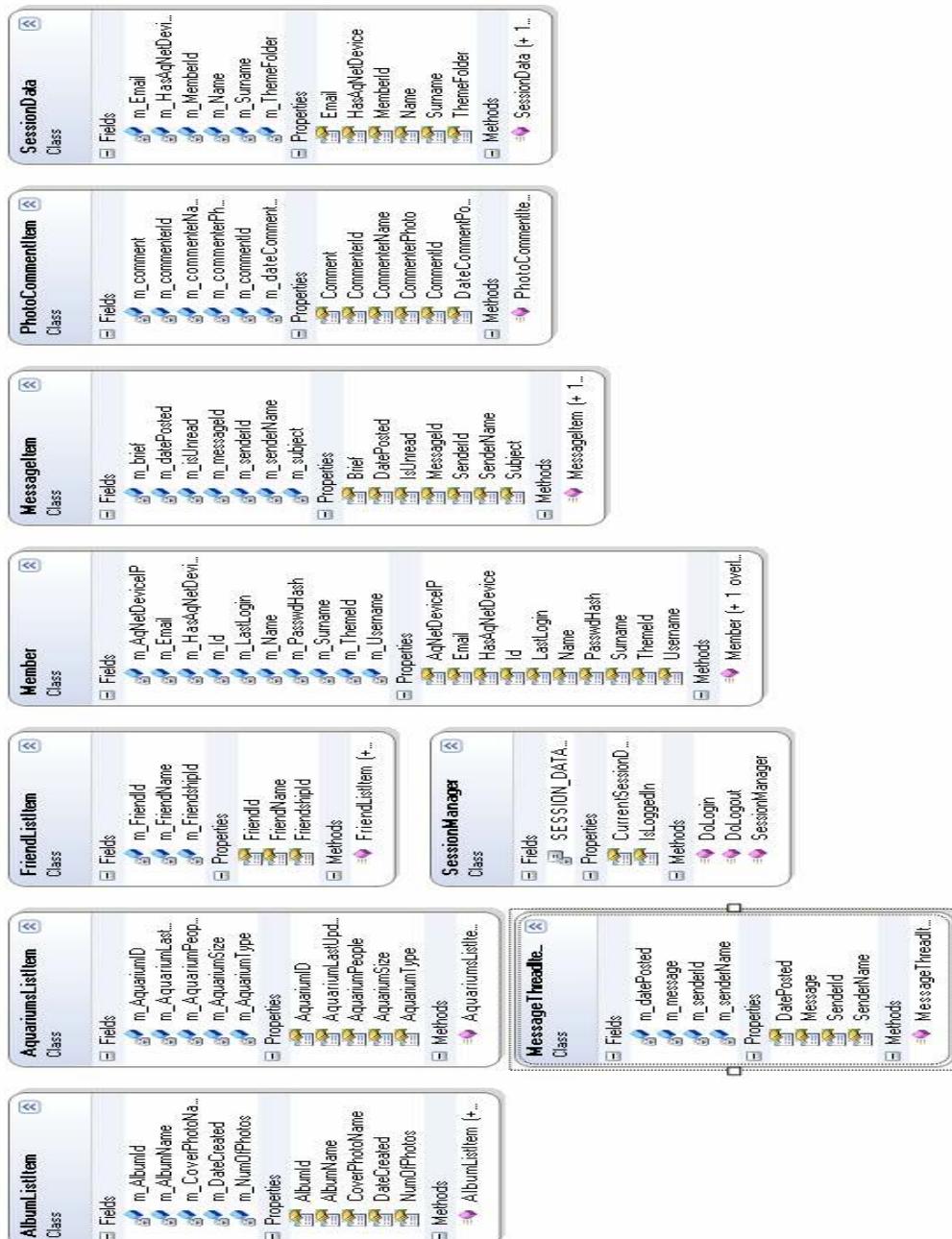


Figure 3.37 – BLL classes diagram

### 3.2.5.1 Classes

**Member.cs:** The class Member comprised of all the general information about the member of the website. The member inserts this information while registering to surfingfish website.

**AquariumListItem.cs:** The class AquariumListItem comprised of all the information (aquarium size, aquarium type, aquarium population) about the member's aquarium.

**FriendListItem.cs:** The class FriendListItem holds the friendship information between members.

**MessageItem.cs:** The class MessageItem holds information about a message members send to one another such as is read? or unread?, date posted and more.

**AlbumListItem.cs:** The class AlbumListItem holds information about every created album. Every user can create album and upload pictures to his page. The information about the picture, such as album name, date it was created, number of photos and more.

**PhotoCommentItem:** The class PhotoCommentItem holds information about album comments. The information about comments, such as date it was posted, name of the commenter and more.

**SessionData.cs:** The class SessionData holds information about the member when member not signed out from the surfingfish website

**SessionManager.cs:** The class SessionManager holds the information about the member's session like log in or log out.

**MessageThreadItem:** The class MessageItem holds information about a message members send to one another.

### 3.2.6 Presentation Layer

#### 3.2.6.1 XHTML & CSS

Designing a modern web site in a standard compliant way means to separate structure from appearance. This means that the HTML code should not contain anything regarding the visual style of the web page, but be written what it was originally meant to do – define structure. To put an emphasis on writing HTML for structure only, XHTML is used instead of HTML. XHTML is basically HTML that is written according to XML rules, which are much stricter than plain HTML. As for appearance – this is where CSS (cascading style sheets) comes into play. This determines how the site will look like and 'feel' to the user. There are several advantages in this separation:

1. XHTML code that defines structure only is much easier to read, maintain and manipulate when needed.
2. A separate object for managing appearance (the CSS) means we can change the entire way the site looks without having to change anything in the XHTML code, and that we can change structure and know how it will look.

**3.2.6.2 Pages** - Essentially, everything comes down to the web pages. They are what the user sees. These are the web pages we created:

**MemeberRegistration.aspx** – This is the first page the user sees after entering the web site's url address. This page asks him for his username/password. If the user isn't registered he comes here, fills out his personal details, and chooses a passwords and registers.

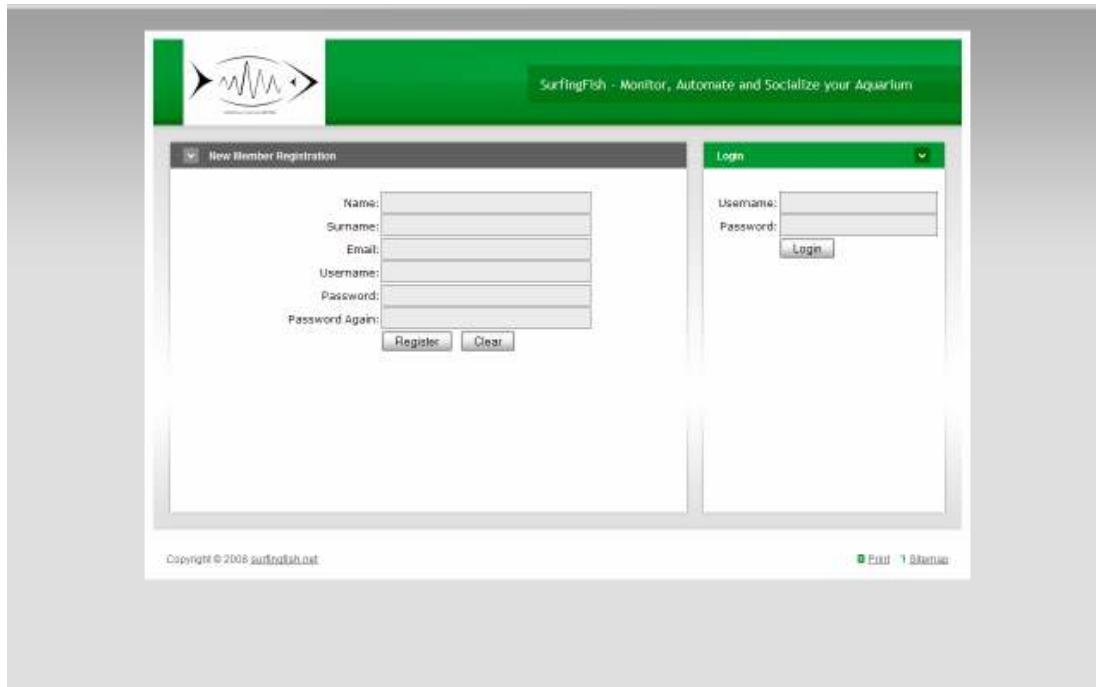


Figure 3.38 – Memeber registration page

**Home.aspx** – This is the first page the user sees after logging in. After logging in the user sees his own main page, and if he selects a friend he sees his friend's main page, pictures aquariums and more.

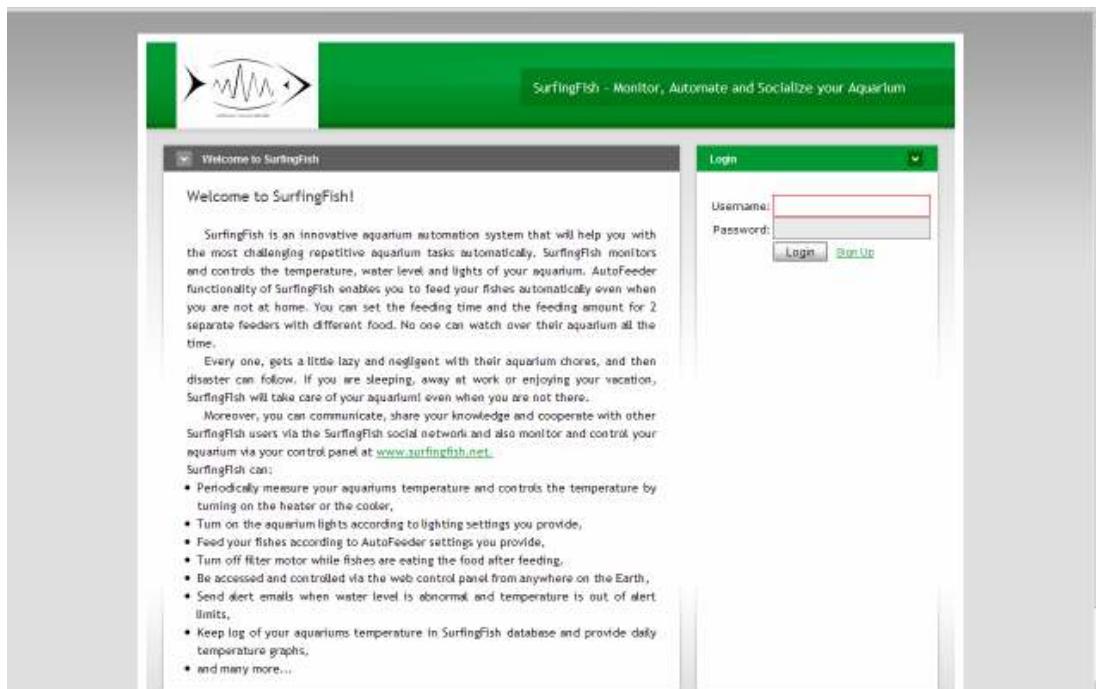


Figure 3.39 – Home page

**Logout.aspx** – If the users want to log out from the surfingfish website they must click the logout tab and redirected to the Home.aspx page.

**Friends.aspx** – This page displays the viewed user's list of friends.

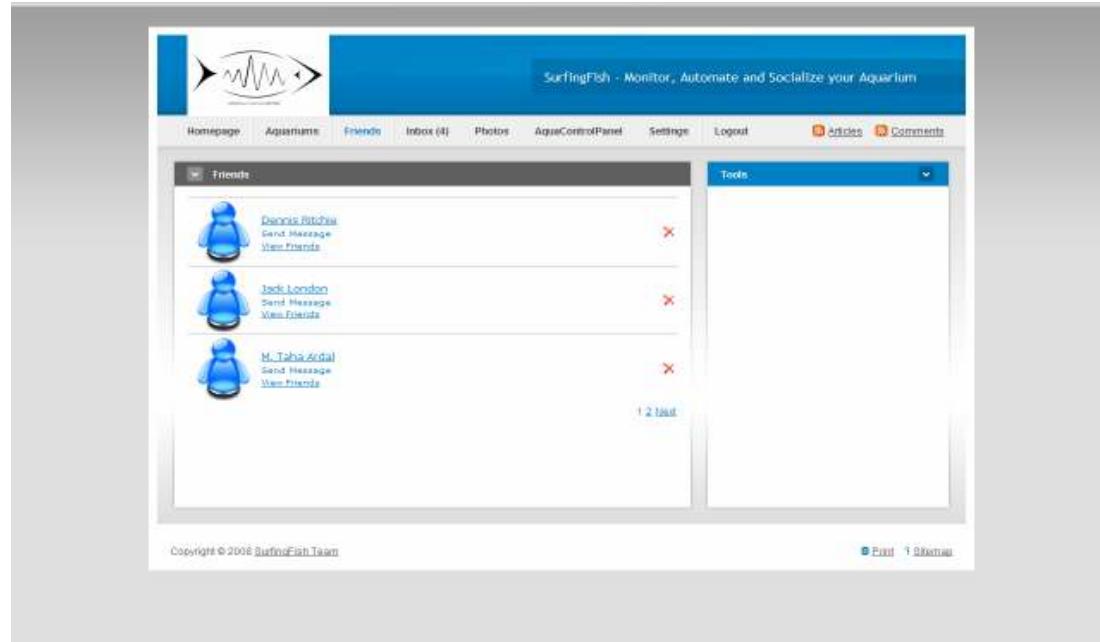


Figure 3.40 –Friends page

**Inbox.aspx** – Displays previews of all the messages that the logged in user received.

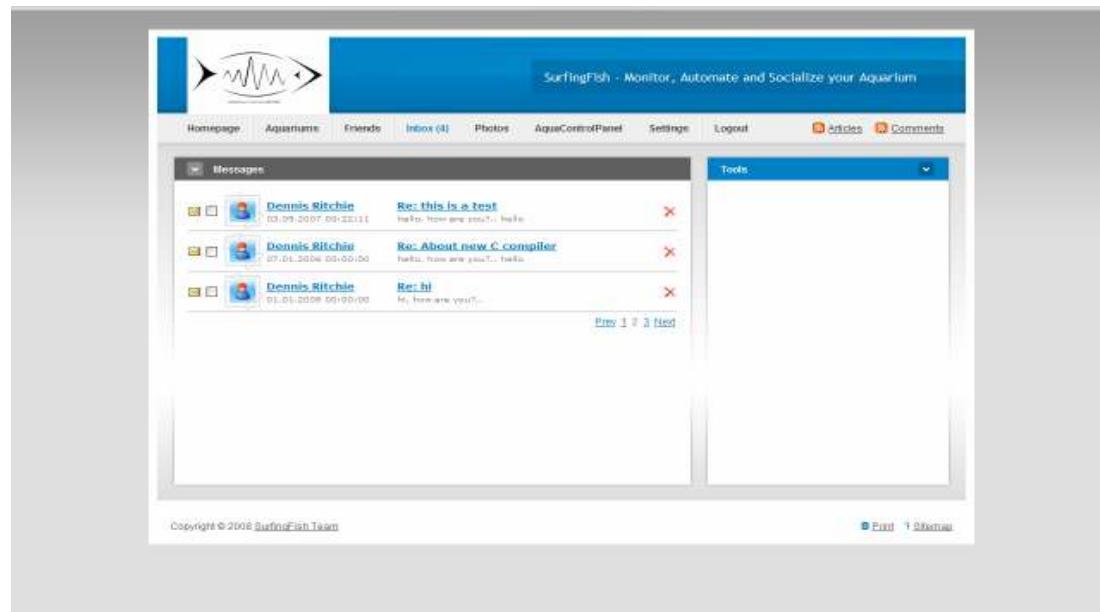


Figure 3.41 –Inbox page

**Photos.aspx** – Here user can watch and upload pictures. Pictures are displayed as thumbnails and the collection of pictures can be paged through.



Figure 3.42 – Photos page

**AquaControlPanel.aspx:** Here user can monitor and control the aquarium parameters.

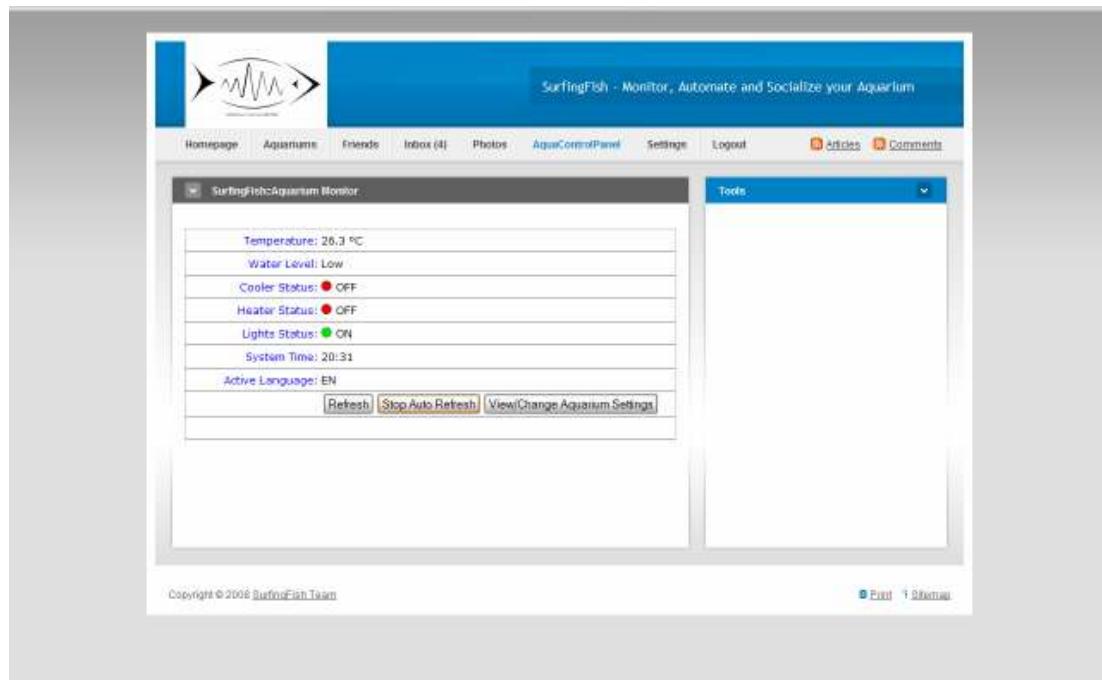


Figure 3.43 – Control panel page1



Figure 3.44 –Control panel page2

**TemperatureStatistics.aspx:** Here you can monitor the aquarium's temprature statistics.



Figure 3.45 –Temperature statistics page

**4 PRICE LIST**

Electronic Components	350 TL
Aquarium Glass	200 TL
Water Cooler	100 TL
Water Level Probe	25 TL
Heater	25 TL
Pleksiglass	50 TL
Pic24 Module	74 TL
Ethernet Module	72 TL
Domain + Hosting	100 TL
Other Aquarium Equipments	90 TL
Gsm Module	71 TL

**Total : 1157 TL**

## 5 CONCLUSION

The project is completed according to proposed features and maintenance issues are now able to be progressed.

Experiments are conducted to examine system limits. Simultaneous inputs are given and the responses of sensors are taken. The project team observed the performances of various equipment and considered about certain replacements.

## 6 TABLE OF FIGURES & TABLES

<i>Figure 3.1 - Project Pier logo</i>	<b>9</b>
<i>Figure 3.2 - Project Pier Page</i>	<b>10</b>
<i>Figure 3.3 - Power supply 3.3V output</i>	<b>10</b>
<i>Figure 3.4 - Power supply 5V output</i>	<b>11</b>
<i>Figure 3.5 - Pic24 general</i>	<b>11</b>
<i>Figure 3.6 – Pic24FJ series</i>	<b>12</b>
<i>Figure 3.7 – Lcd Screen</i>	<b>12</b>
<i>Figure 3.8 – Keypad controller</i>	<b>13</b>
<i>Figure 3.9 – Keypad interface mechanism</i>	<b>13</b>
<i>Figure 3.10 – Keypad working</i>	<b>13</b>
<i>Figure 3.11 - Relays</i>	<b>14</b>
<i>Figure 3.12 - Filtering. Upper view of Aquarium</i>	<b>15</b>
<i>Figure 3.13 - Temperature Control</i>	<b>16</b>
<i>Table 3.1 - Power VS Required Time to Change a 70 lt. Aquarium's Temperature by 1° C</i>	<b>16</b>
<i>Figure 3.14 - Water Cooler System</i>	<b>17</b>
<i>Figure 3.15 – Temperature sensor</i>	<b>17</b>
<i>Figure 3.16 - Water Level Monitor and Possible Alarms</i>	<b>19</b>
<i>Figure 3.17 – Feeder side view</i>	<b>20</b>
<i>Figure 3.18 – Feeder top view</i>	<b>20</b>
<i>Figure 3.19 – Reel time clock</i>	<b>21</b>
<i>Figure 3.20 – ENC28J60</i>	<b>22</b>
<i>Figure 3.21 – Ethernet board</i>	<b>22</b>
<i>Figure 3.22 – Old system box</i>	<b>23</b>
<i>Figure 3.23 – First version of jacks</i>	<b>24</b>
<i>Figure 3.24 – Second version of jacks</i>	<b>24</b>
<i>Figure 3.25 – Jacks after modified</i>	<b>25</b>
<i>Figure 3.26 – 3D model of system box</i>	<b>25</b>
<i>Figure 3.27 – New system box</i>	<b>26</b>
<i>Figure 3.28 – SurfingFish board</i>	<b>26</b>

<i>Figure 3.29 – Ajax and http diagram</i>	<b>34</b>
<i>Figure 3.30 – MySQL logo</i>	<b>35</b>
<i>Figure 3.31 –Visual Studio</i>	<b>35</b>
<i>Figure 3.32 – Sql Manager for mysql</i>	<b>36</b>
<i>Figure 3.33 – 3-Tier approach</i>	<b>37</b>
<i>Figure 3.34 – Database diagram</i>	<b>38</b>
<i>Table 3.2 – Members table</i>	<b>39</b>
<i>Table 3.3 – Friendship_request table</i>	<b>39</b>
<i>Table 3.4 – Friendship table</i>	<b>39</b>
<i>Table 3.5 – Messages table</i>	<b>40</b>
<i>Table 3.6 – Reply_messages table</i>	<b>40</b>
<i>Table 3.7 –Albums table</i>	<b>41</b>
<i>Table 3.8 – Album_commnets table</i>	<b>41</b>
<i>Table 3.9 – Photos table</i>	<b>43</b>
<i>Table 3.10 – Photo_comments table</i>	<b>42</b>
<i>Table 3.11 – Profiles table</i>	<b>42</b>
<i>Table 3.12 –Themes table</i>	<b>43</b>
<i>Figure 3.35 – Stored procedure diagram</i>	<b>43</b>
<i>Table 3.13 – Friends stored procedure table</i>	<b>44</b>
<i>Table 3.14 – Memebers get all stored procedure table</i>	<b>44</b>
<i>Table 3.15 – Activate account stored procedure table</i>	<b>44</b>
<i>Table 3.16 –Authenticate member stored procedure table</i>	<b>45</b>
<i>Table 3.17 – Insert pending stored procedure table</i>	<b>45</b>
<i>Table 3.18 –Get message stored procedure table</i>	<b>46</b>
<i>Table 3.19 – Unread messages stored procedure table</i>	<b>46</b>
<i>Table 3.20 –Recieved messages stored procedure table</i>	<b>46</b>
<i>Table 3.21 – Reply messages stored procedure table</i>	<b>47</b>
<i>Table 3.22 – Album name stored procedure table</i>	<b>47</b>
<i>Table 3.23 – Get albums stored procedure table</i>	<b>47</b>
<i>Table 3.24 – Get photo stored procedure table</i>	<b>48</b>
<i>Table 3.25 –Photo comments stored procedure table</i>	<b>48</b>
<i>Table 3.26 – Insert photo stored procedure table</i>	<b>48</b>

<i>Table 3.27 –Insert photo comment stored procedure table</i>	<b>49</b>
<i>Figure 3.36 – DAL classes diagram</i>	<b>49</b>
<i>Figure 3.37 – BLL classes diagram</i>	<b>51</b>
<i>Figure 3.38 – Member registration page</i>	<b>54</b>
<i>Figure 3.39 – Home page</i>	<b>54</b>
<i>Figure 3.40 – Friends page</i>	<b>55</b>
<i>Figure 3.41 – Inbox page</i>	<b>55</b>
<i>Figure 3.42 – Photos page</i>	<b>56</b>
<i>Figure 3.43 – Control panel page1</i>	<b>56</b>
<i>Figure 3.44 – Control panel page2</i>	<b>57</b>
<i>Figure 3.45 – Temperature statistics page</i>	<b>57</b>

## 7 REFERENCES

- [1] R.E. Yalçın. “Akvaryum Serüvenine İlk Adım”. **Makeleler(2005)** [www.akvaryum.com](http://www.akvaryum.com)
- [2] R.E. Yalçın. “Akvaryum ve Suyun Kimyası”. **Makeleler(2005)** [www.akvaryum.com](http://www.akvaryum.com)
- [3] [www.projectpier.org/about/](http://www.projectpier.org/about/)
- [4] T. Kuloglu. “Biyolojik Filtrasyon”. **Makeleler(2005)** [www.akvaryum.com](http://www.akvaryum.com)
- [5] <http://en.wikipedia.org/wiki/Asp.net>
- [6] <http://en.wikipedia.org/wiki/Ajax>
- [7] <http://www.mysql.org>
- [8] [http://en.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio)
- [9] <http://sqlmanager.net/>

## 8 CONTACTS

### PROJECT TEAM

**Name Surname:** ENDER PİYALE

**University:** KADIR HAS UNIVERSITY

**Faculty:** ENGINEERING

**Department:** ELECTRONIC ENGINEERING

**Email:** enderpiyale@gmail.com

**Name Surname:** AHMET ARDAL

**University:** KADIR HAS UNIVERSITY

**Faculty:** ENGINEERING

**Department:** ELECTRONIC ENGINEERING

**Email:** ardalahmet@hotmail.com

**Name Surname:** ÇAĞRI İLBAN

**University:** KADIR HAS UNIVERSITY

**Faculty:** ENGINEERING

**Department:** ELECTRONIC ENGINEERING

**Email:** cagriilban@yahoo.com

### PROJECT SUPERVISOR

**Name Surname:** ASSIST. PROF. DR OSMAN KAAN EROL

**University:** İSTANBUL TECHNICAL UNIVERSITY

**Faculty:** ELECTRICAL – ELECTRONIC ENGINEERING

**Department:** COMPUTER ENGINEERING

**Email:** okerol@itu.edu.tr