Mobile Trainer Fitness Management System

By

Jason Phillips

Submitted to the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

June 2005

Acknowledgments

First, I like to thank my mother, Deborah Phillips, for her prayers and emotional support. I would like to thank Sean Carr and Shannon Trannor for helping me test my project. I also want to express my gratitude to Professor Robert Schlemmer, my academic advisor, for his feedback and support of my project.

Abstract

Mobile Trainer is a complete fitness management solution that integrates PocketPC mobile devices and Windows Desktop PCs. Mobile Trainer was designed to address the need for a reliable and inexpensive fitness management solution. This project focused heavily on programming, database functionality, and web server administration. This system was programmed in c# and Visual Studio .Net Enterprise 2003. MySQL was implemented as the back-end database that stored fitness exercises, history, and user information. The web server chosen was Cassini.

This project uses a distributed two-tier system. The first tier is the client front end that can reside on a PocketPC device or a Windows desktop machine, version 2000 or above, machine. It was as developed with c# to communicate with the second tier Web Service.. The second tier consists of the actual Web Service application, IIS, and the MySQL open-source relational database. It encapsulates the mySql connectivity, retrieval, and storage functionality. The total commercial cost for development of the project was $1,749.22.

Table of Contents

List of Figures

Mobile Trainer Fitness Management System

1. Statement of the Problem

1.1. Background

At the time of this project, PocketPC Personal Device Assistants (PDAs) were on the horizon of dominating the PDA device market at 40% market share (1). The key advantage of PocketPC over PDA competitors such as Palm was capability. PocketPC devices were designed to harnesses the power of faster mobile device processors and improved networking with 802.11b.

Although PocketPC devices offered greater functionality, there were still a limited number of software solutions at this time, particularly for fitness management. Those fitness management solutions designed to run on a PocketPC Operating System were overpriced and lacked proper data recovery if the mobile device battery died based on data gathered in my initial research. *

- **Please refer to Appendix A for a review of alternate solutions for PocketPC fitness management.**

1.2. Statement of Need

I believe there was an opportunity to create a fitness management solution that could be used in home gyms and fitness centers to address the issues of PocketPC in regards mobile fitness applications for PocketPCs.

2. Product Description and Intended Use

2.1. What is Mobile Trainer?

This project focused on developing a scalable Fitness Management Solution (FMS) for Mobile Devices and Desktop PCs, called Mobile Trainer. The primary areas Information Technology used to develop were in Web Application Programming and Database design concepts.

Mobile Trainer is an all-in-one fitness management solution designed to help manage exercise plans and meet fitness goals. It provides an on the fly installation scheme and incorporates an extensive tool-set to motivate users, improve fitness tracking, and member-to-trainer communication.

2.2. Intended Use

Typically users, or those who physically exercise, would track their fitness goals through the Mobile Trainer front-end on a PocketPC enabled device or a PC running a Windows Operating System (OS). Trainers would use the Windows Desktop interface to manage and interpret user's data and develop custom workout plans. Regardless of users' fitness goals, he or she could have a customizable work-out plan. Exercise instructions and pictures could be loaded from the database if needed. In addition, Mobile Trainer would allow users and trainers to analyze their workout history and generate meaningful reports on the fly.

2.3. User Profiles

The user profiles for Mobile Trainer consisted of three major profiles:

- Fitness Member
- Trainer or Home User
- Administrator

This fitness client role was designed for the average user that exercises in the gym. They would use the PocketPC client to track their information in most instances. His or her personal account and profile would be set up by a Trainer or an Administrator.

A trainer was considered to be an individual who creates fitness goals for the gym members. The fitness trainer could also exercise using the Mobile Trainer application. Internally, the FMS had set up a trainer role similar to the typical user with additional administrative functions such as adding users, creating exercise plans, viewing overall performance charts.

Administrators had entire control over the Mobile Trainer FMS. In most cases, the Administrator did not have an active role in Mobile Trainer on a day-to-day basis. Administrators would only need to provide an account with sufficient privileges for mobile trainer to set-up Cassini and run the mySQL service.

2.4. Design Protocols

Mobile Trainer was designed using .Net and incorporates web services to generate XML dynamic content. At runtime, a proxy server is generated and binds XML data to objects. The figure below depicts a high-level overview of how the Mobile Trainer system functions.

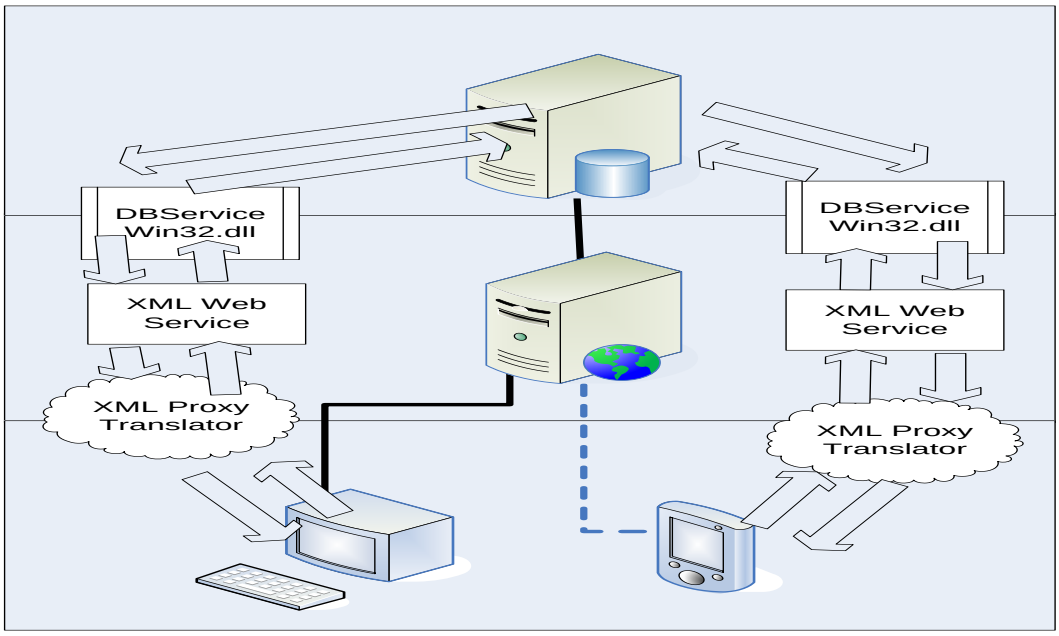

Figure 1. Technical Overview

The following database tables comprised Mobile Trainer from a Data Architecture standpoint:

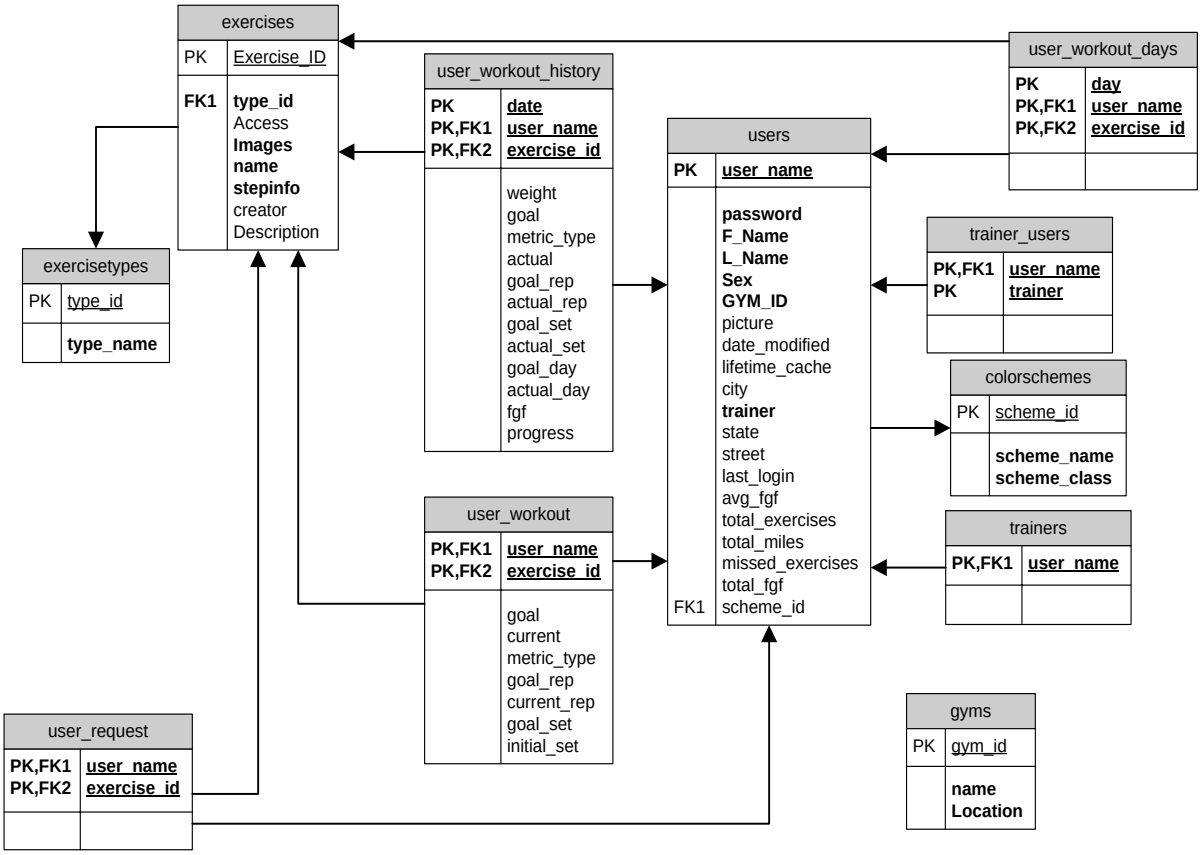

Figure 2. Database Schema of MobileTrainer

There were two front-end applications for this system. One was the PocketPC client. Fitness Members would primarily use this client front-end to manage his or her workout that day. The second front-end was the desktop client. This client application could run on any Windows Operating System with the .Net framework. The Desktop client not only offered the same functionality as the PocketPC device, it also offered fitness administration tools for trainers. The user or trainer could have some freedom to customize the look and feel of the User Interface in regards to the font and background color on the desktop client. The figure below highlights what users were saw when using the PocketPC and Desktop clients:

3. Deliverables

The list below outlines the deliverables that were accomplished in this project:

- Create Client PC Front-end (C# Smart device)
- PC interface front-end interface for Trainers and Users
- Fitness Tracking Tools Developed for Fitness Members.
- Exercise Sheet
- History Information
- Exercise Request
- Server Configuration Utility
- Approval Workflows created for Trainers
- Report generation and chart setup completed.
- Export user data to optional format. (On PDA and PC Client)
- Offline synchronization capability for PocketPC
- Web XML Service Developed and Tested
- mySQL create database to store fitness information
- Installation package that installs necessary software
- Create help interface and documentation

4. Resource Management

4.1. Budget

Name	Manufacturer	Cost	Explanation	Type
Visual Studio .Net Enterprise	Microsoft	$1799.00	Development Environment	Software
Adobe Photoshop CS	Adobe	$649.00	Develop Graphics	Software
Intel Pentium 4 w/ Win XP	Custom Built Bundle	$320.26	Test Device to Host Server	Hardware
Toshiba E-740	Toshiba	$249.99	Test Device (PocketPC)	Hardware
USB Switch	Omega	$29.99	Wireless Testing Device	Accessory
MySQL	MySQL AB	**FREE**	Backend Database	Software
Tomcat	Apache	**FREE**	Web Server	Software
Grand Total		$1,749.22		

Table 1. Budget

4.2. Timeline

Senior Design 1 Task	Start	Finish	Complete	In Progress	Not Started
1. Research Idea	9/26/04	11/19/04	✓		
2. Progress Report 1	10/21/04	10/21/04	✓		
3. Progress Report 2	11/1/04	11/4/04	✓		
3. Oral Proposal	11/16/04	11/18/04	✓		
4. Written Proposal	11/4/04	12/1/04	✓		
Senior Design 2 Task	**Start**	**Finish**	**Complete**	**In Progress**	**Not Started**
1. Setup Hardware	10/28/05	12/19/05	✓		
2. Progress Report 1	1/20/05	1/20/05	✓		
3. Progress Report 2	2/20/05	2/20/05	✓		
5. Design Freeze of Draft	2/6/05	2/10/05	✓		
6. Database Development Complete	1/4/05	2/25/05	✓		
7. XML Transformation API completed	1/4/05	1/17/05	✓		
6. PC Client Completed	1/4/05	2/25/05	✓		
7. Pocket PC Complete	1/4/05	2/25/05	✓		

7. Testing	1/18/05	2/12/05	✓		
4. Initial Prototype	1/16/05	2/20/05	✓		
8. Technical Presentation	3/4//05	3/10/05	✓		

Senior Design 3 Task	**Start**	**Finish**	**Complete**	**In Progress**	**Not Started**
1. Analyze Faculty Feedback from Senior Design 2	3/1/05	3/20/05	✓		
2. Progress Report 1	3/18/05	3/18/05	✓		
3. Progress Report 2	3/30/05	3/30/05	✓		
4. Training Documentation	4/1/05	5/20/05	✓		
5. Required Writing	3/12/05	3/19/05	✓		
6. Final Presentation	5/1/05	6/02/05	✓		

Table 2. Timeline

4.3. Software and Hardware Rationale

Visual Studio 2003 Enterprise was chosen because it offered a more efficient application development environment. Other Integrated Development Environments (IDEs) such as Java's Net Beans or Eclipse required a significant amount of preparation before coding could begin. The time required to deploy an XML web server was significantly less in Visual Studio because developers were not required to modify xml configuration files and other mundane task that are required to set-up Java Web Services.

Photoshop was chosen because of the ability to use layering and create custom graphics. I was also comfortable with Graphics application because of past project experience and training at school.

mySQL stood out as the best Database solution at the time of the project. There were several advantages to using mySQL. mySQL was an open source product. Therefore the overhead involved with managing licensing was non-existent. In addition it offered features comparable to commercial databases such as foreign key constraints with innodb, blob, and relational integrity.

5. Proof of Design

The next section discusses, in detail, how the deliverables of the project were fulfilled and what challenges where encountered during the development process.

5.1. Client Interfaces

The front-end interface for PocketPC was created with the required functionality for users to store their fitness data during a work out session. The front-end interfacse on the desktop client included all the functionality of the PocketPC it also includes advance functionality for Administrators.

5.2. Fitness Tracking Tools for Fitness Members

The fitness tracking tools were embedded onto a single form. The tracking tool had intrinsically two objectives. One was its design to provide all the information a user needed to enter his or her data in for a fitness exercises. The other objective was to help instruct members how to correctly use an exercise.

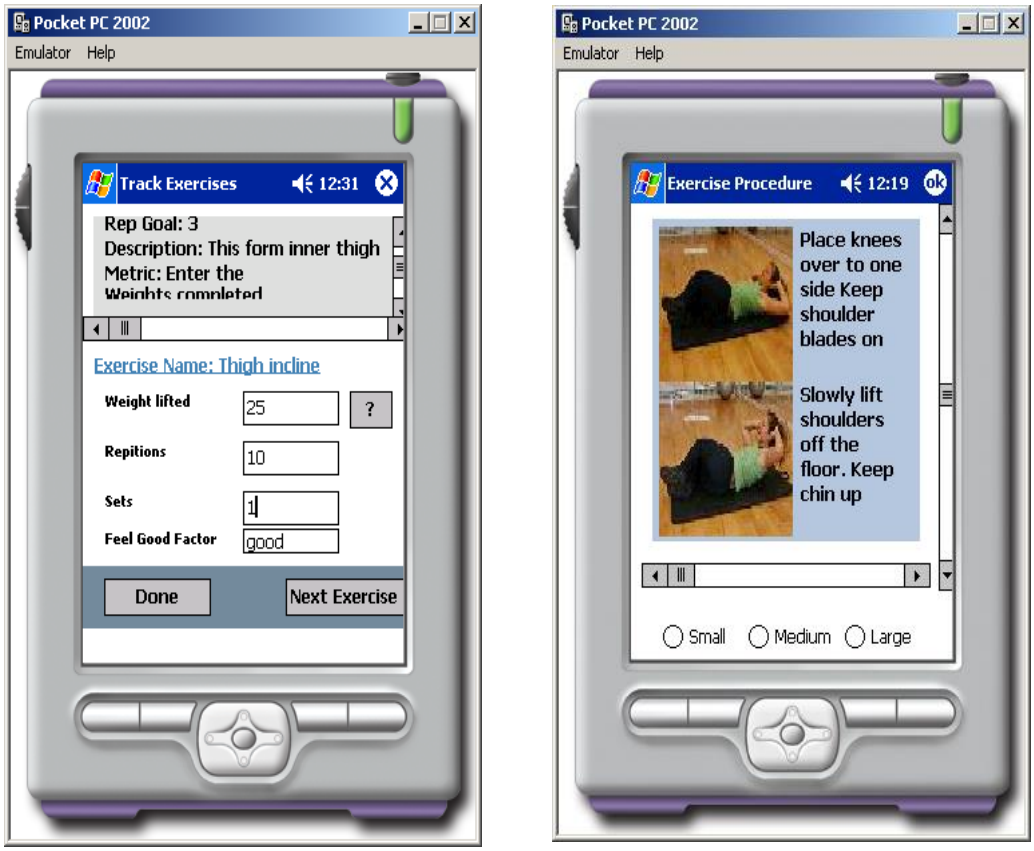

Figure 3. Exercise Tracking Tools

5.3. Exercise Sheets

Trainers could generate exercise sheets for a fitness member if a situation would arise such as PDA not being available. It could be used to as a tangible reference that a user could look at to determine when he or she was scheduled to workout.

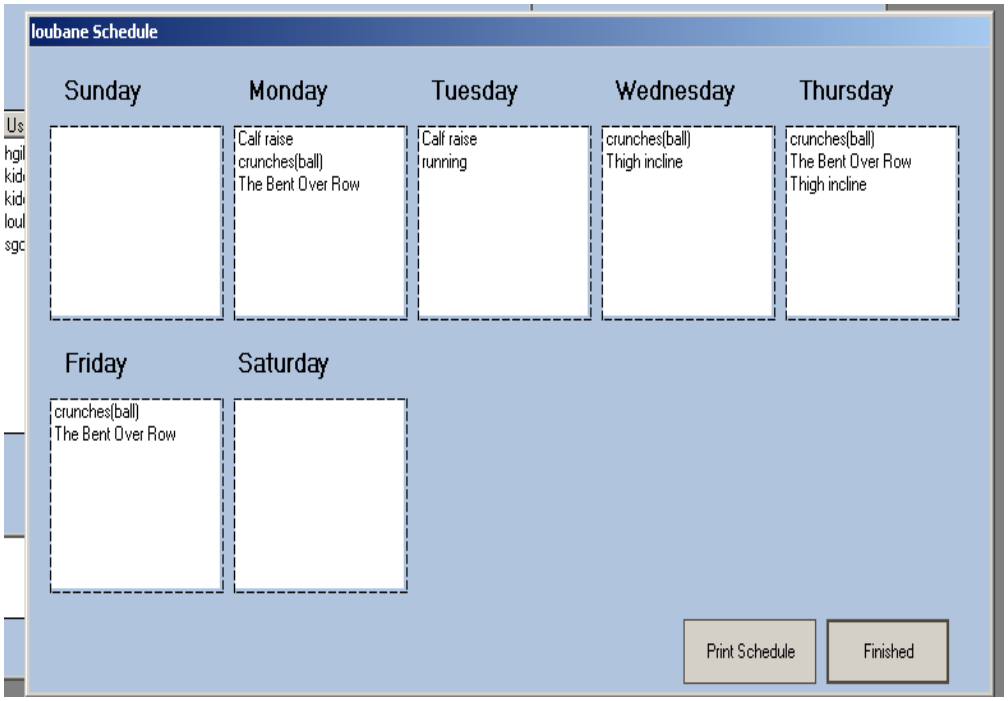

Figure 4. Exercise Schedule Form

5.4. Automatic Installation Package

The mobile installation package could used to install the front-end interfaces or back-end Web Service. Administrators would start the installation wizard and decide which application to install. The figure below highlights the installation wizard utility.

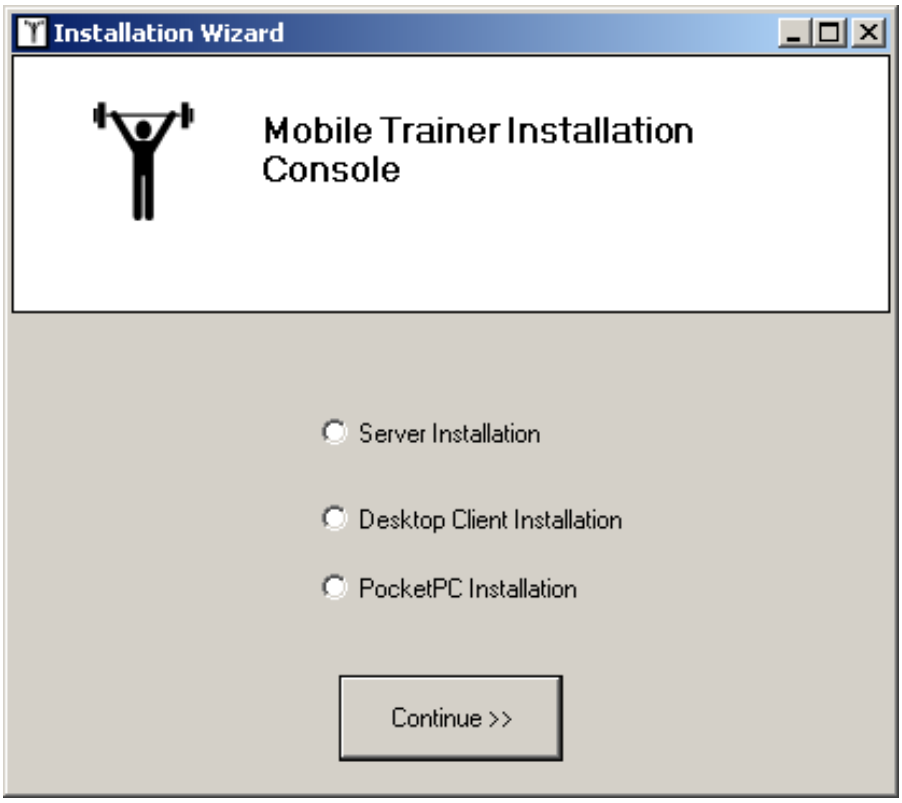

Figure 5. Mobile Trainer Installation Wizard

5.5. History Information

Fitness members can see their history information in three ways. One way is the through the ‘view history’ button. Another way is to export their data to excel using the export tool which had capability to be used by trainers.

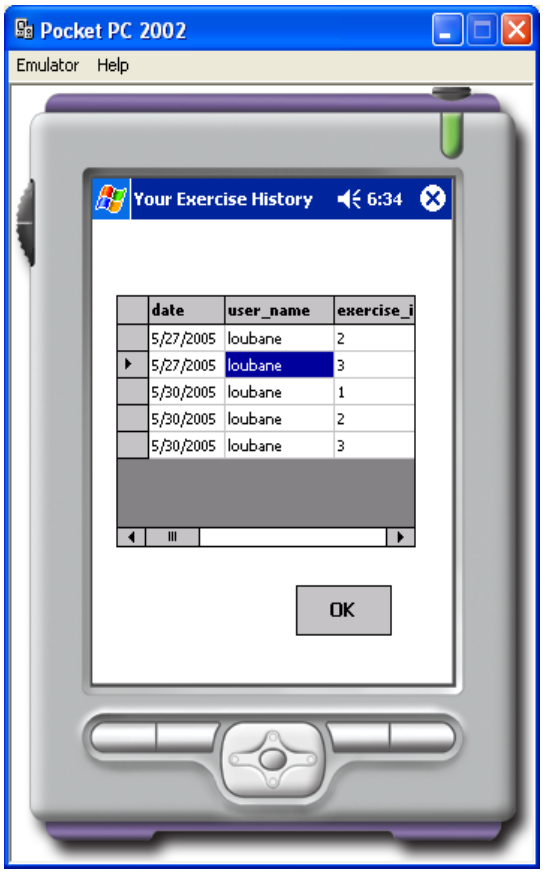

Figure 6. View History Form

5.6. Exercise Request

Exercise requests were fairly easy to implement. The fitness member simply had to click the tools tab on the PocketPC client and highlight choices the exercises he or she wanted a trainer to add.

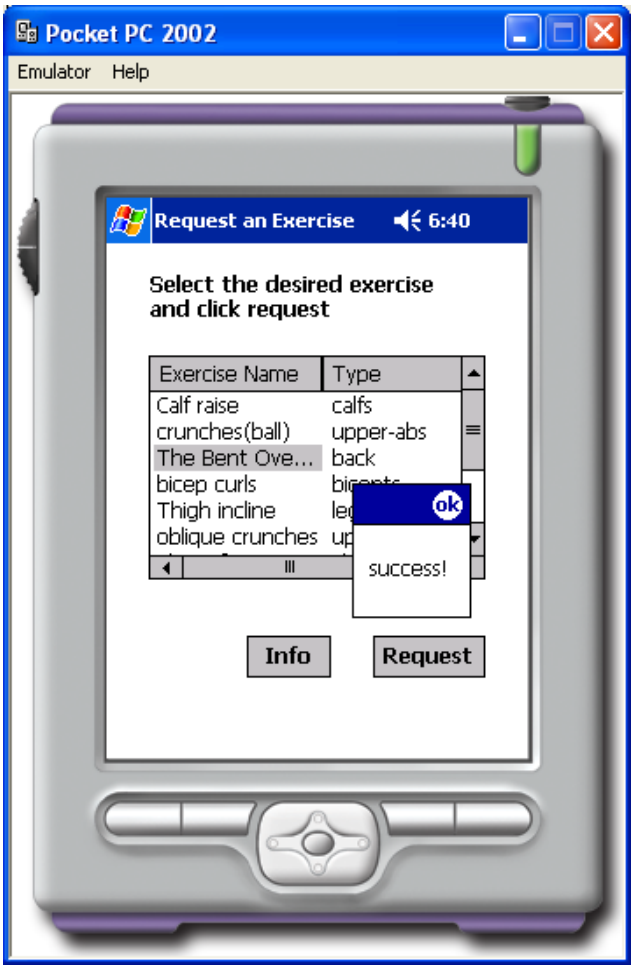

Figure 7. Exercise Request Form

5.7. Server Configuration Utility

The Server Configuration Utility met the desired outcome by allowing a centralized source to start and stop the Cassini web service. The server configuration utility was small but also effective.

Figure 8. Server Configuration Utility

5.8. Chart Generation Tools

Users who were fitness members or fitness trainers had chart generation tools at his or her disposal. Fitness members could use the chart progress tool to align with his or her personalized fitness goals.

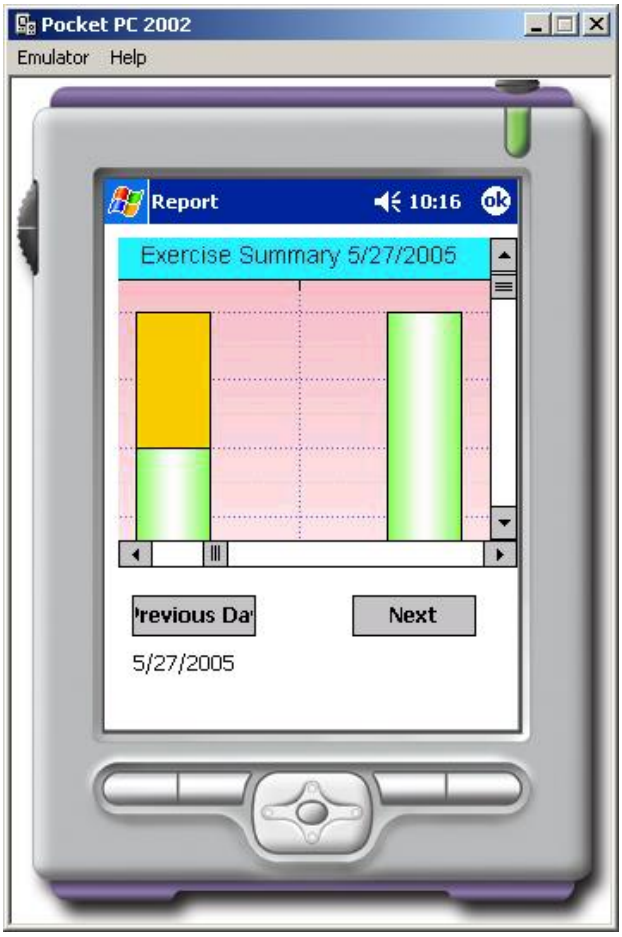

Figure 9. Chart Review Tool Pocket PC

Fitness Trainers had two types of charts. One chart could determine the overall Feel Good Factor (FGF) of each fitness member a trainer had. This would allow a fitness trainer to determine how effective he or she was in creating users work-out plans.

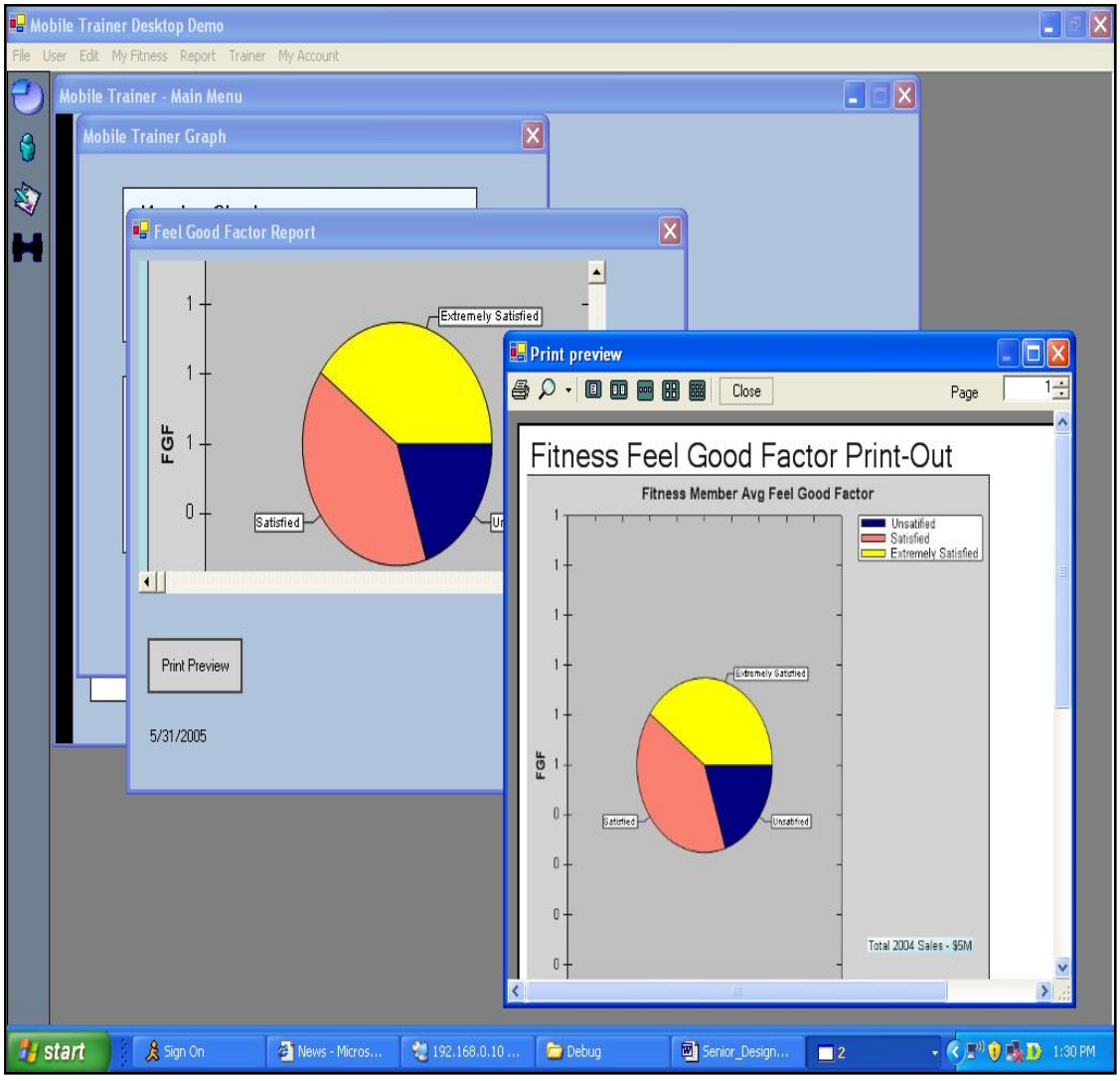

Figure 10. Desktop Chart Review Tool

5.9. Export Capability

A key deliverable in this project was the capability to export exercises data to an excel format. Fitness members and trainers could easily export his or her data in both front-ends.

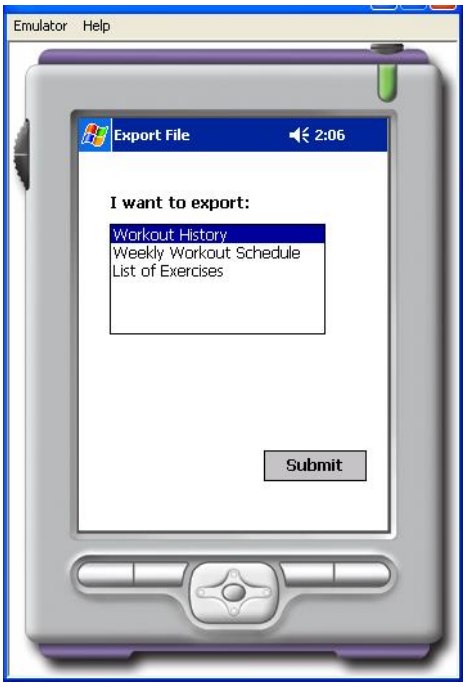

Figure 11. Excel Export Form

5.10. Offline Synchronization for PocketPC

Offline synchronization was possible through the PocketPC mobile client. Users could enter his or her fitness data regardless of whether he or she was connected to the Web Server. Offline data allowed for faster response time without the loss of information while a user was offline.

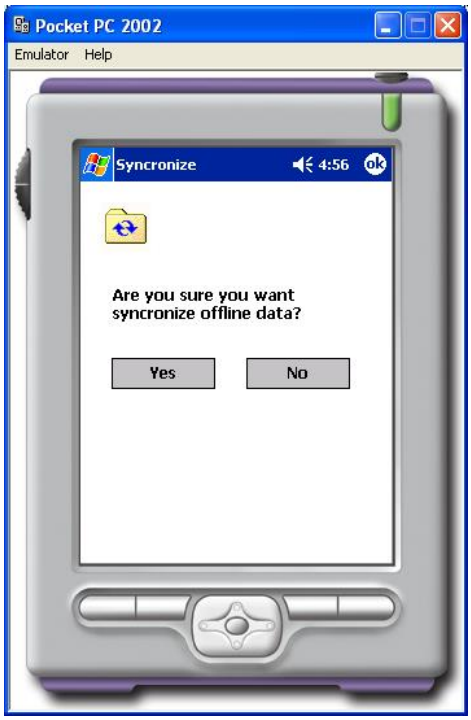

Figure 12. Synchronization Form

5.11. Web XML Service Developed and Tested

The web service in this project used a DLL to that communicated to the mySQL database and returned the results based on the function a Client system involved. I enjoyed learning about web services and their application in distributed environments. The web service and DLL may have been the time the most consuming development for this project.

5.12. Database Implementation

The database implementation was successful. mySQL contained all relevant user, trainer, exercise, and historical data in regards to the Fitness Management System. Except for the exercise information, the mySQL schema was backed up under a backup file that would allow administrators to have a restore point if the Database became corrupt.

5.13. Help Interface and Documentation

Fitness Trainers and fitness members could easily find the information they needed with the help documentation packaged with mobile trainer. The PocketPC and Windows Desktop clients have similar help documentation. The figure below shows an example of the help interface for Mobile Trainer.

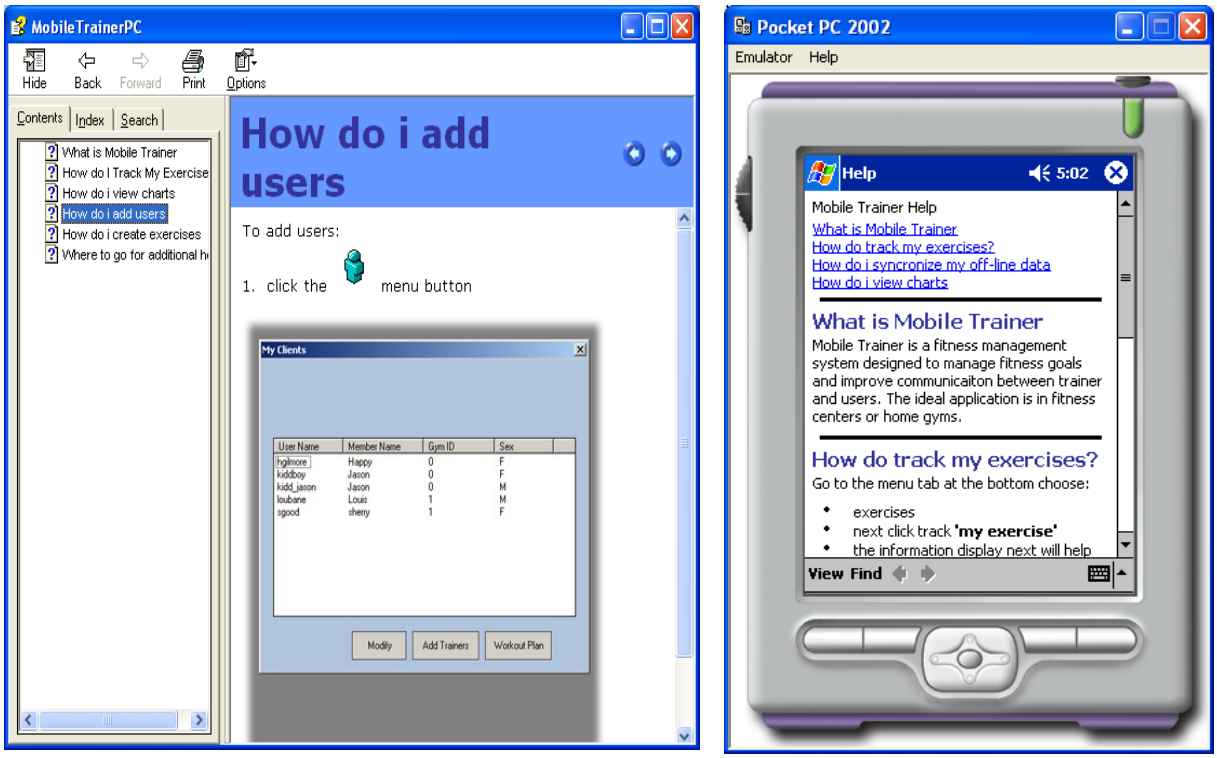

Figure 13. Help Documentation for Desktop Client and PocketPC Client

6. Conclusion and Recommendations

6.1. Conclusions

Mobile Trainer Fitness Management System was designed to help fitness advocates reach his or her fitness goals. At the time of this project, there were very few fitness management tools for PocketPC. Those solutions on market did not offer data recovery

features. Mobile Trainers addressed these issues and provided a marketable solution to home users and gyms.

6.2. Recommendations

This project gave me exposure to various programming technologies. However, I would recommend those who would like to develop a similar multi-tier architecture take into consideration factors of development. One factor to consider is time. The average time I spent on development ranged from 40 to 60 hours each week. I attribute a lot of this time to deploying the PocketPC emulator when changes were made. Another consideration is the time required set-up and learn web service technology. Web service technology was fairly straight forward in .Net but, it still took weeks of reading to understand how to implement it as a solution in my project.

Appendix A

There are a lot of mobile software applications and clones designed to accommodate the needs of our faced based mobile environment. The table below summarizes the list of top five products. Notice there is only one client for pocket PocketPC at the time.

Product Name	Company	Platform	Cost	Features
Nutribase	CyberSoft, Inc.	Palm OS, Windows	$249.99	▪ Advance nutrition calculation ▪ Health database ▪ Add work outs ▪ Hot Sync to PC ▪ Reports ▪ Sort Data ▪ Assignment Profiles ▪ Multiple Users
Cross Trainer	Cross Trainer ©	Palm OS, Windows	$49.99	▪ Manage Nutrition ▪ Add Work Outs ▪ Track Workouts ▪ Fitness Reports
MySport Training	Exposè Corp	Pocket PC	$29.99	▪ Sort Data ▪ Track Workout ▪ Synchronize Data
Personal Training Workstation	WORKOUTWARE .COM LLC	Palm Os, Windows	$29.99	▪ Fat Calculator ▪ Track workout ▪ Progress Charts ▪ Weight Progress
PDA Fitness	Handmark™	Palm Os	$29.99	▪ Add bench settings ▪ Multiple Users ▪ Sort Data ▪ Track Workout ▪ Synchronize Data

Appendix B

Code Snippets

B1. Store Dataset Snippet

During development, I needed an effective solution to update any table in the database. The following snippet is from the shared library that is encapsulated inside the Web Service method updateDataSet():

```
public bool UpdateDataSet(DataSet ds, string table_name)
            {
                  DataSet userDS = new DataSet();
                  MySql.Data.MySqlClient.MySqlConnection conn = new
MySql.Data.MySqlClient.MySqlConnection("server=127.0.0.1;user id=root;
password=trainer; database=Trainer; pooling=false");
                  conn.Open();
                  MySql.Data.MySqlClient.MySqlCommand command;
                  MySql.Data.MySqlClient.MySqlDataAdapter mysqlDa = new
MySql.Data.MySqlClient.MySqlDataAdapter("select * from " + table_name,
conn);
                  //mysqlDa.MissingSchemaAction =
MissingSchemaAction.AddWithKey;
                  MySql.Data.MySqlClient.MySqlCommandBuilder cb = new
MySql.Data.MySqlClient.MySqlCommandBuilder(mysqlDa);



                  if(mysqlDa.Update(ds,table_name) > 0)
                  {
                        return true;
                  }
                  else
                  {
                        return false;
                  }

                  conn.Close();


            }
```

B2. Extract Exercise Instructions Snippet

The following code snippet addressed the issue with how to deserialize a serialized array object in the database. Since the .Net Compact Famework does not

support deserialization of array objects, I came up with a solution to extract each binary image and instruction into a table which could be read on the PocketPC device**.** The method definition is listed below:

```
public DataSet getInstructions(DataRow dr)
      {
            byte[] bt = (byte[]) dr["Images"];
            byte[] bt2 = (byte[]) dr["stepinfo"];
            DataSet ds = new DataSet();

            Image[] img;

            MemoryStream ms = new MemoryStream(bt);
            BinaryFormatter bf = new BinaryFormatter();
            img = (Image[]) bf.Deserialize(ms);


            ms = new MemoryStream(bt2);
            string[] instr = (string[])bf.Deserialize(ms);
            ms.Close();

            DataTable dt = new DataTable("Instruct");
            dt.Columns.Add("Image");
            dt.Columns.Add("Instruction");

            dt.Columns["Image"].DataType = bt.GetType();
            dt.Columns["Instruction"].DataType =  (new
            String('c',1)).GetType();

            for(int i = 0; i < img.Length; i++)
            {
                  DataRow new_row = dt.NewRow();
                  Bitmap bp = new Bitmap(img[i]);
                  MemoryStream im = new MemoryStream();

                  img[i].Save(im,
                  System.Drawing.Imaging.ImageFormat.Jpeg);
                  new_row["Image"] = im.ToArray();
                  im.Close();
                  new_row["Instruction"] = instr[i];



                  dt.Rows.Add(new_row);
            }
            ds.Tables.Add(dt);
            return ds;


      }
```

References

1. .NET Compact Framework. April 24, 2003. Retrieved November 3, 2004 from http://msdn.microsoft.com/chats/transcripts/mobileembedded/embedded_041703.aspx

2. Cordes, Kyle. ".NET Compact Framework - Development Considerations." http://www.kylecords.com. 2004

3. Dotnetjunkies.com. http://www.dotnetjunkies.com/Weblog. 2005

4. Ferrara, Alex. Programming .*Net Web Services*. California. O'reilly Press. 2002.

5. Legard, David. "Pocket PC Gaining Ground in PDA Market". http://www.pcworld.com/news/article/0,aid,106463,00.asp. 2002

6. Martin. James. "Mobile Computing: Palm Vs. PocketPC." http://www.pcworld.com . 2003.

7. Microsoft. Microsoft.com. 2005

8. MySQL. MySQL Reference Manual.http://www.mysql.com. 2005.

9. Nayak, Girish. "Write details to Excel file using Excel Object." http://www.dotnetspider.com. 2005

10. Vallinayagam, Iatha. "Pocket PC Calling Dynamic Web Service". http://www.codeproject.com. 2005.