



Fowler Software Design

Intranet Connector

Team:
Jon Kreuger
Branden Salis
Brett Shibao
Phillip Suitt

Abstract

The task Fowler Software Design requested was to improve functionality and continue development on existing software, *The Link*. *The Link* is a user interface that connects the company to a database of information such as activity reports of technicians, client records, a history of projects and compensation, etc... All of the data is stored in a hierarchically structured database containing many tables created in a mix between the legacy FoxPro and SQL programming languages. The database uses Primary and foreign keys within tables to link all of the data together. By using SQL queries and stored procedures *The Link* can be updated to contain more valuable information such as deductions and compensation percentages.

The requirements include a conversion between two databases (FoxPro to Unified), the development of new reports that can be given to technicians and clients, and an improvement upon *The Link*. The project is necessary to help with client and company relations by giving the staff access to more information that has either not been created, or is still in the legacy FoxPro database. Some of the additions to *The Link* include: new compensation report printing (converted from FoxPro to Unified) with additional improved invoice printing, and the ability to view and add deductions and/or a list of requirement activities per tech.

The problem will be broken down into smaller subtasks and will be completed in parts. Tasks such as database conversion can be separated from the updating of the actual web interface; one is done in Microsoft SQL while the other is done in C#. Two languages allow the project to be broken up into parts and summed up into a final product. The SQL part of the project can be broken down even further into single stored procedure subtasks. In order to convert the FoxPro reports to be printed on the Unified side, there must first be many stored procedures that can all be worked on separately. Though both are necessary for the final production, each individual section can be tested with test data, to ensure each part will work when the parts are combined, and to reduce the necessity of one part being completed in succession.

Table of Contents

I. Requirements Specification.....	2
A. Introduction.....	2
B. Project Description.....	2
C. Requirements.....	2
i. Functional.....	2
ii. Non-Functional.....	3
D. Scope.....	3
E. Use Cases.....	3
II. System Design.....	5
A. Design Goal.....	5
B. High Level Design.....	5
C. Design Details.....	6
D. Implementation.....	9
III. Conclusions.....	15
A. Summary.....	15
B. Future Directions.....	15
IV. Glossary.....	16
V. References.....	17
VI. Appendix.....	18
A. The SQL Databases.....	18
B. The Report Manager.....	22
C. <i>The Link</i>	24

I. Requirements Specification

A. Introduction

Fowler Software Design (FSD) is a company that develops software based on the specific needs of its clients. The company mainly develops software for the Microsoft operating system. By using the Microsoft SQL programming language, FSD is able to organize all of their company's information into a very organized, yet complex system of data tables. *The Link* is a web interface developed by FSD that displays the data tables as a user-friendly web interface which allows for the easy viewing, changing, and printing of all the company's information. This project will provide more advanced development for the user-interface.

B. Project Description

The objective of this project is to further advance the development of FSD's web interface, *The Link*. FSD has two databases; one that uses FoxPro/SQL (Legacy FoxPro) and one that only uses SQL (Unified). The databases are a compilation of all the information for the company which includes: staff members, clients, projects, invoices, deductions, etc... The software that FSD wanted developed is an intranet connector for all of this database information. The connector will need to have improved functionality for showing more information that has or has not been created. The major problems that were addressed are conversion of the current system, Legacy FoxPro, to a system which is only SQL, and implementing the functionality of the new database, Unified, to *The Link*. The major benefit of this project is that they will be able to more efficiently access and communicate data stored in the database.

C. Requirements

Currently there are two databases (Unified and FoxPro) which provide information for *The Link*. The code for modifying *The Link* will involve two languages: C# and ASP.NET. C# will be used for implementing functionality in *The Link* and ASP.NET web forms will be used for interface formatting. The code for the databases will be SQL and FoxPro.

i. Functional

- Results of the database queries must output as reports
- Reports will then be printable
- Both databases can be used to create a combined report
- Make Web Interface access more parts of the database

- Allow staff to do everyday tasks more easily
- Error messages will be displayed when information being saved is not complete

ii. Non-Functional

- Conversion from FoxPro/SQL to SQL
- Create reports using SQL queries and Visual Studio Report Manager
- Code will be added in the current language, C#
- Documentation will be made for users of *The Link*
- Web Interface integration onto different platforms

D. Scope

- Introduced new tables and data types to the database which in-turn created new reports
- Implemented all of the new reports to *The Link*
 - Buttons for printing reports
 - Functionality for addition to or modification of the database
 - Credit Memos
 - Deductions
 - Compensation Percents
 - Project Leaders
 - Practice Managers
 - Technicians
 - Practice Managers
- Fixed reports in which output was incorrect
- Partial conversion from FoxPro/SQL to only SQL

E. Use Cases

E.1 - User Opens *The Link* using <https://uds.fowlersoftware.com>

Description: Web Interface (*The Link*) opens up and displays a login screen for staff members

Pre-condition: None

Post-condition: *The Link* opens up displaying a login screen

Primary Flow:

1. Program information is gathered and the interface opens as a website
2. Displays login screen

E.2 - User logs into *The Link*

Description: Gathers information from database and displays the user's available resources as a webpage

Pre-condition: *The Link* is at the login screen waiting for user input

Post-condition: Interface opens showing the user's available resources

Primary Flow:

1. User inputs ID and password
2. Database is checked to see if these field values are valid
3. Displays the user's available resources on *The Link*

Alternate Flow:

- 2.a. User is not found in the database

Alternate Flow:

- 3.a. "Invalid Login" displayed on the interface

E.3 - User prints List of Requirements

Description: User wants to print the "List of Requirements by Category" report for a specific project.

Pre-condition: Interface is displaying user's available resources

Post-condition: Prints out report "List of Requirements by Category"

Primary Flow:

1. User clicks "Projects" button to see a list of current projects
2. User clicks "Print Report" for the project he/she wants to print the report for
3. Data is gathered from the database
4. Data is put into the pre-made report layout and printed

Alternate Flow:

- 3.a. Data is unable to be gathered
- 4.a. Report prints but is void of data

E.4 - User prints a list of their current activities

Description: User wants to print a list of their current activities/requirements for their projects/clients.

Pre-condition: Interface displays user's available resources

Post-condition: “List of Requirement Activities” is printed

Primary Flow:

1. User clicks on activities button to display the requirement activities for that user
2. User clicks a print button to print the “List of Requirement Activities”
3. Data is gathered from the database
4. Data is put into a premade layout and is printed

Alternate Flow:

- 3.a. Data is unable to be gathered
- 4.a. Report prints but is void of data

E.5 - User inputs what they did for the day

Description: User wants to log what they did for the day and on which activity

Pre-condition: Interface displays user’s available resources

Post-condition: The user’s input is stored in the database and will show up on the interface until the current week is finished.

Primary Flow:

1. User clicks on time tracker button to display the time logged for that week
2. User selects from a dropdown box a particular activity which is part of a particular project
3. User inputs what was done for that activity during the time interval put in.
4. User finishes and data is sent into the database and is displayed on the interface

Alternate Flow:

- 3.a. Time interval is incorrect or rate per hour is incorrect
- 3.b. Interface displays an error at the top of the page.

II. System Design

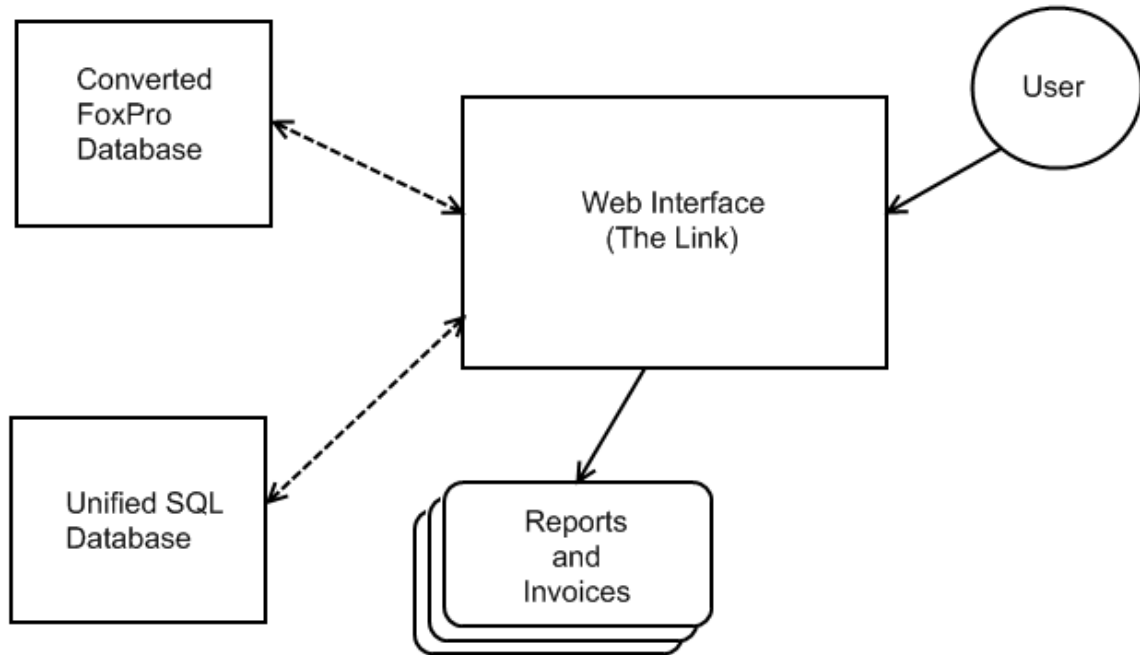
A. Design Goal

The goal was to design documents for clients or staff members of Fowler Software Design and then to implement the documents on *The Link* so that the reports are easily accessed and may be printed when needed. Also, the goal included the capability to allow *The Link* to accept and store or edit the data in the database.

B. High Level Design

The current interface is set up to provide the staff the ability to:

- Document their hours and what they did during them
- Print out invoices and reports for specific projects/clients
- Keep track of particular projects/requirements

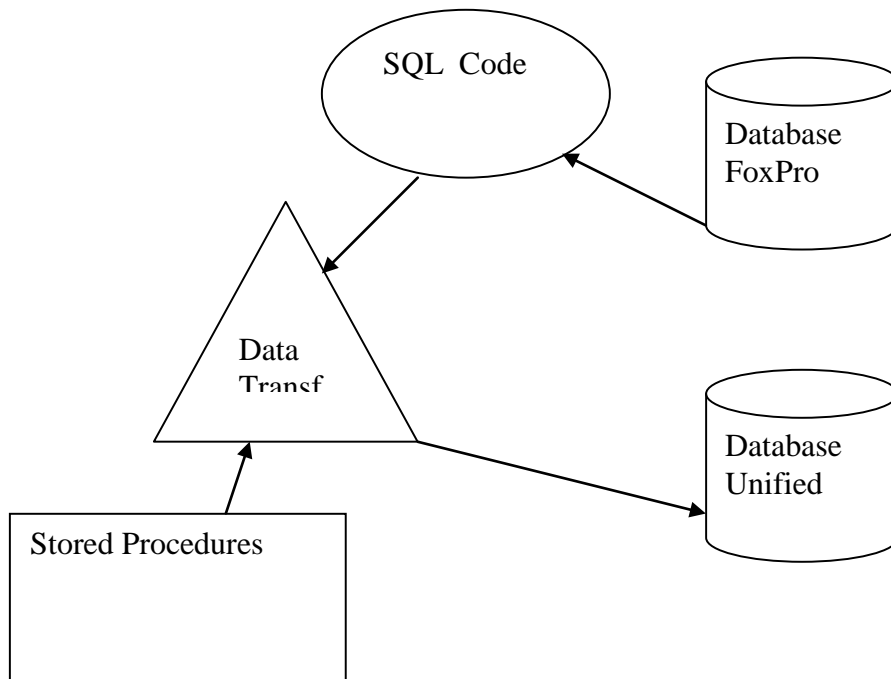


C. Design Details

Modules

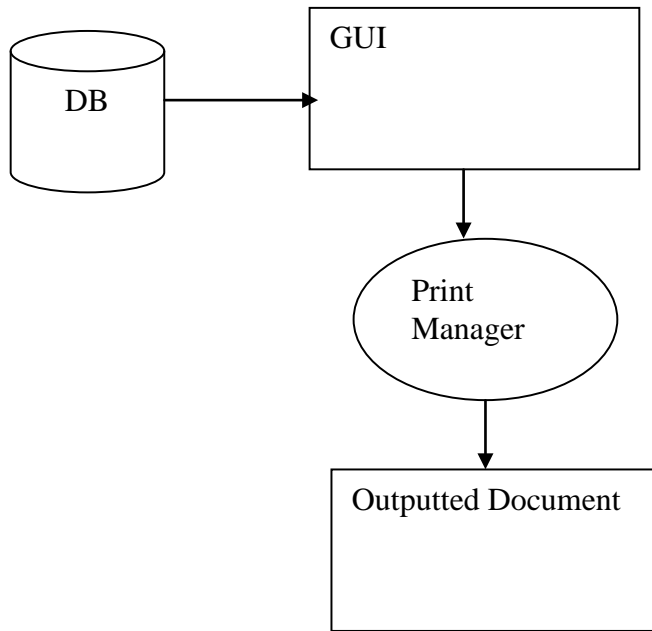
C.1 Data Transfer Module

This module handles the accessing of the Databases. This is composed of SQL stored procedures which will handle the data transfer from FoxPro to a Unified Database. It will also handle the retrieval of data from the databases, to allow the client to access requested data. This will take place prior to the client accessing the GUI. See the figure below for a diagram of the data transfer.



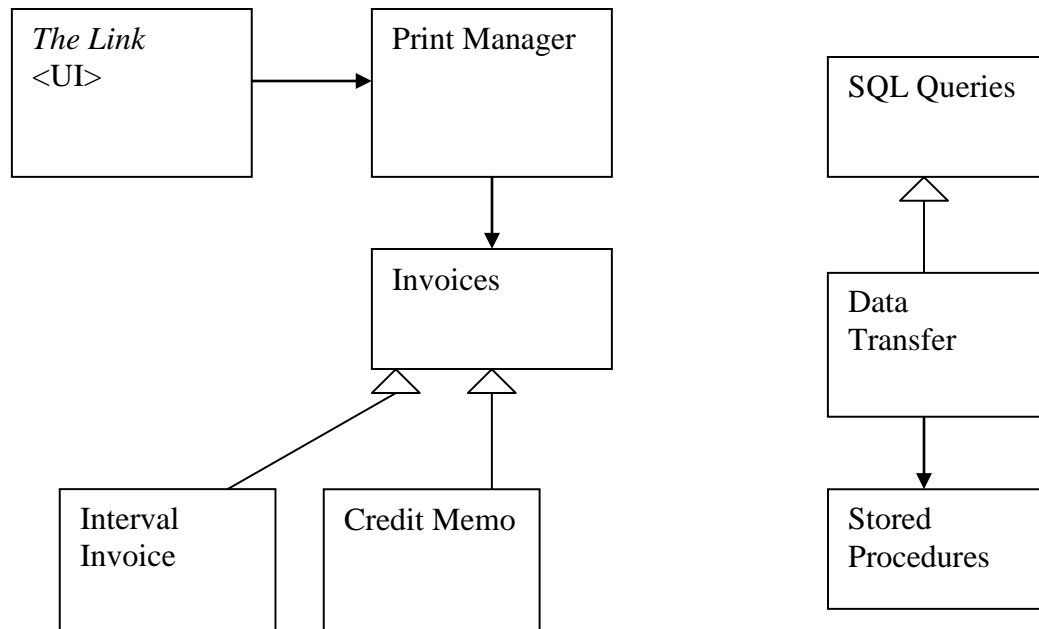
C.2 Reporting Module

An interface used by the client to access the database after the data transfers have occurred. It will allow the client to print documents, quickly and easily. It is also known as *The Link*, and is a web based interface to allow users to access it anywhere internet is available. See the figure below for a diagram of the reporting module.



C.3 UML Diagram

The figure below is a rough UML of how the program will interact.



C.4 Database Schema

The following is a schema for the final program. The program will pull data from tables to create a report to be printed. All the tables are in one of two databases, FoxPro and Unified. Every table has a primary key to help keep track of the corresponding data in the other tables.

Public

public_id: primary key
public_name: name of the client

Project

project_id: primary key
public_id: foreign key

Requirement Category

requirement_category_id: primary key
requirement_category_name: lists the name of the requirement category

Requirement

requirement_id: primary key
requirement_name: lists the name of the requirement

Activities

activity_id: primary key
requirement_id: foreign key
staff_member_id: foreign key to a staff member who will accomplish the activity

Staff Member

staff_member_id: primary key
first_name: lists the first name of the staff member
last_name: lists the first name of the staff member
salary_per_hour: decimal value of staff member salary per hour

D. Implementation

The Project includes 3 main locations in which code and other elements are implemented. Those sections are:

1. The SQL Databases
2. The Report Manager
3. *The Link*

D1.1 The SQL Databases: Overview

Within the existing SQL Databases (those containing the Unified data as both The Normal Database [NORMAL] and The Warehouse Database [WAREHOUSE] collectively [Databases], and the automatically created database which contains the FoxPro data [LEGACY]) among the multiple Tables containing the actual numbers and bits comprising the data, many pieces of coding exist whose only purpose is to automatically work with the Tables. These pieces of code are known as Stored Procedures. The reasons for the existence of these Stored Procedures are for many purposes, but ultimately in the overall schema, they are there to make editing and maintaining the Databases far simpler and easier for the outside user. For instance, calculating and adding 4-10 rows per week to a Table with 2 given values and 20 other values each calculated from other Tables in the Databases by hand and entering those values manually is a monumental task not only for efficiency, but also for data accuracy. Thus, Stored Procedures are implemented to make certain changes and large scale updates or creations to the Databases with very little user effort and possible error.

D1.2 Stored Procedures

Stored Procedures fall into a variety of categories depending on what their actual effect is on the Databases. The categories experienced in the Project are as follows:

- Insertion
 - These Stored Procedures specifically create new rows within Tables in the Databases either from user input or from automatically selected input given user requirements.
- Updating
 - These Stored Procedures specifically update row values which already exist in the Databases either from user input or from automatically selected input given user requirements.

Insertion Stored Procedures appear in the Databases with a frequency relatively close to the number of automatically generated or updated Tables in the Databases. This is because for every Table, you need only one Stored Procedure to add entire rows to the Table. Having more than one Stored Procedure for any given Table, leads to inefficiency on the user's part, and user erroneous input is more likely to occur. On the other hand, Updating Stored Procedures can have a frequency per Table of one Stored Procedure per each row in that Table. Most Tables have one or a few rows which are updated only during the Insertion Stored Procedure for that Table, but the rest of the rows are automatically generated from existing data within the Databases. In some cases, the relation between individual column values within in a single row are so similar in calculation, it is more efficient to update these values within a single Stored Procedure instead of one Stored Procedure per value.

D1.3 Database Locations

FSD maintains the Databases between two separate and distinct computers on their network. One of these machines is called Development, and the other is called Production. The Production machine is the instance of the Databases which is always “Live” or in other words, “the real copy” of the Databases. Any input from *The Link* or reports printed from The Report Manager, obtain and input data through Production. On the other hand, Development is periodically updated as a whole from a direct copy of Production. Thus any accidental deletions or mistakes made while maintaining the Databases on Development, will not affect the company and can easily be replaced by another copying of Production to Development.

In light of this distinction between Development and Production, a programmer working on improvements to the Databases and/or Stored Procedures needs a fail-safe method of taking his work from Development to Production without error. The way this is done at FSD is through a shared network folder called LINK_UPDATES. Within this folder, the programmer will save any and everything he does on Development in order to replicate exactly those changes onto Production. Stored Procedures and Tables cannot be trusted to be manually updated on Production by the programmer due to the possibility of error even when copying from a saved file in the shared folder. Thus, any changes to Development which are to be saved on Production are saved in the shared folder as Scripts.

D1.4 Scripting Changes

These Scripts are run within the SQL Server environment and automatically replicate the changes they were created to reproduce. There are Scripts for many purposes, some of those purposes are as follows:

- Altering Stored Procedures
 - These Scripts change the particular Stored Procedure which already exists in the destination Databases.
- Creating Stored Procedures
 - These Scripts will add a new Stored Procedure which did not previously exist on the Production machine, but the programmer recently created for a specific purpose on the Development machine.
- Altering Tables
 - Unlike most Stored Procedure Scripts, these Scripts are automatically generated by the SQL environment upon the saving of a Table which was changed manually by the programmer on the Development machine. This means that Table already existed in the Databases before the edits made by the programmer, but the changes he made needed to be updated on the Production machine as well once they were complete.
- Creating Tables
 - Often, new features or calculations are needed within the Databases and the frequency of the calculated data values does not exist in any of the currently existing Table. This calls for a new Table with foreign keys relating it to the

Tables from which it gets its frequency of data. Thus, the creation of this Table will be vital for whatever purpose it serves and the programmer must make sure it receives a Script to then update Production with the new Table.

- Altering Views
 - Within the Databases, a concept of Views exists to make the obtaining of data from many Tables who are all “sub-Tables” of a primary Table. These sub-Tables all have a foreign key to only the primary Table, and have the same frequency as the primary Table and can all be viewed in the same format without discrepancy between data values. In such cases, a View is especially relevant. When obtaining data from many of these hierarchical Tables, the programmer would have to implement joins between all his multiple select statements to make sure all the data fit together properly. The View takes care of all those joins internally to display all the same-frequency data in one place to simply retrieving the data for later use.
 - In the case where any Table in the Databases is updated, the corresponding View must also be updated so that any calls to the data within the updated Table can be called directly from the corresponding View. These Scripts will change the existing Views for a class of Tables to include the new changes the programmer made to the underlying Tables.
- Creating Views
 - Similar to the Altering of Views when the View's underlying Table is changed, a brand new Table will not have its own View until the programmer creates one on Development. Thus, these Scripts are those changes to the Databases to include the new View for the new Table.
 - With this information about Table and View Scripts, one may determine that it is impossible for accurate updating on Production if there are Create or Alter Scripts for either a Table or a View within the LINK_UPDATES folder, but no corresponding Create or Alter Script for the opposing Table or View.

D1.5 Example

See VI.A.1 for example Script. The given Script is the largest Script created during the Project. It is very large due to temporary Table creation and multiple sums over intervals within the Databases in order to align the proper data from the LEGACY database. The overall goal of this Script is to create the Stored Procedure

`create_invoice_exception_driver` which in turn creates a Table called `foxpro_vs_unified_exceptions`. This table contains data to determine the individual programmer and client for which he worked as well as the value of service delivered (VSD) for that particular project sorted by `week_id` overall. In essence, this single stored procedure generates all the data needed by the Report Manager to run the Exception Report to show the discrepancies between the FoxPro and Unified calculations.

D2.1 The Report Manager: Overview

The Report Manager is really the in-between step connecting The SQL Databases (Databases, see D1.1) with *The Link*. For instance, it would be highly inefficient to have the C#/ASP.NET (see D3.4) coding of *The Link* configure and create reports based off the Databases every time a user on *The Link* needed a report on certain data within the Databases. Running reports in this manner would cause horrendous amounts of server lag and user impatience at the extended length of time for report generation. The solution: The Report Manager

D2.2 How it works

The Report Manager is actually run on a separate computer from Development and Production (which hosts *The Link* directly). Using Visual Studio 2005 with the report Manager addition, reports are able to be designed and published from Development to the Report Manager machine. Once published to the remote machine, the Report Manager can run them on current live data from Production. When users on *The Link* query for a particular report to be shown, or to print reports with or without viewing them on screen, *The Link* sends a simple request to the Report Manager on the separate machine to create the requested report. Once created on the machine with relatively low loads in comparison to Production running *The Link*, the user views it through *The Link*. In this way, the creation of Reports does not add to the overall workload of the Production machine.

D2.3 Getting Data for Reports

The Report Manager gets the data for each individual Report by a SQL query within the Report file itself. This query generates a Table within the Report Manager from which the Report can then be created. Reports need to be generated as quickly as possible for the sake of user patience and computer workload. The longest amount of processing time associated with any report, will always be obtaining the data from a remote host. Thus, the query for obtaining the data should be as short and simple as possible; it shouldn't have many or any internal calculations. This is another reason Stored Procedures are beneficial to the Databases and the efficiency of how The Report Manager gets data for the reports. For most Reports, the data only changes once per week, not every time the Report is created. Thus, running a single Stored Procedure at week end to generate all Tables used in all Reports, will save time over the course of the week. If Stored Procedures are always created for any Report, then Report creation will always be fast and efficient as it is only a single `select * from *` and any necessary joins to merge the Tables based on their frequencies.

D2.4 Designing the Reports

Reports have to look nice when presented to the user. This is the bottom line for any visual display of information to the end user in any scenario. Thus, design of the Report, once the proper data is being gathered, is essential. Visual Studio includes a auto-

compiled design with the programmer's supplied sorting and grouping of the garnered data. But this auto-compiled layout is never adequate for a decent Report design. So every Report must be hand-designed by the tech to adhere to the exact output specifications needed by the user. Once the design is perfected by the programmer, he will publish the entire Report to The report Manager for use from an external location such as *The Link*.

D2.5 Example

See VI.A.1 for a Report of the Project. Obviously this Project was unpaid work for FSD and thus all mentionable values contained therein are 0, but the variety of Categories and Requirements can be seen along the left side of the Report.

D3.1 *The Link*: Overview

The Link is the entire GUI website through which FSD operates. The Project (see I.B) included updating *The Link* to “make life easier” for FSD staff and their Programmers by adding features and new Reports to help deviate FSD from the reliance on the FoxPro software.

D3.2 Features

The Link has many features and they all facilitate the accounting and time tracking between all the Programmers and Clients at FSD. Without *The Link*, the weekly close outs would be arduously difficult to complete in one working day by the staff members who perform all the financial duties every week.

D3.3 Permissions

The Link includes permission levels for all the users who have the ability to access it. Without permissions, any user could tamper with the internal workings of FSD and accidentally or purposely cause fallacies in the Databases and accounting done at FSD. Thus, the permissions given out by FSD to its individual Programmers is vital to maintaining the integrity of the Company's internal workings. Though *The Link* seems simple and plain at first glance, it is actually very central to the workings of FSD.

D3.4 Coding

The Link is coded in two different ways. The first is the design of the pages themselves including the layout of the items on each page. This code is done using ASP.NET language and methods from within Visual Studio 2005/2008. The other piece of coding that goes into the design of *The Link*, is the C# coding behind every ASP.NET design page. The C# code controls what each item of the ASP.NET page will contain from the Databases. To save user-entered information into the Databases, the C# coding is used to

either directly input the data, or more efficiently, run a Stored Procedure with the given input as variables to the Stored Procedure. Both the coding languages need to be used for each and every page of *The Link* in order for the pages to work.

D3.5 Example

See VI.C.1 for the User Guide to *The Link*. This Guide contains all elements which are necessary to using *The Link* on a daily basis from both the average Programmer entering his time, to the FSD Secretary closing out the week and printing all of that week's invoices.

III. Conclusions

A. Summary

The Project succeeded with high remarks not only from the FSD manager of the Project, but also from those FSD members awaiting the new product. All requirements given by the Client were met and in some cases, exceeded. The requirements given by the Colorado School of Mines (CSM) were also met and in some cases, exceeded. This Project was also difficult in some instances to fit into the predefined layout of CSM's expected project layout, but the obstacles were overcome and the presentations completed.

The Programmers in this Project, Jon Krueger, Branden Salis, Brett Shibao, and Philip Suitt (Collectively, "the Team"), all completed collaboratively on every task assigned by the Client. The Team has also determined that their individual efforts throughout the Project were equal. In addition, for the furthering of their knowledge and the opportunity to participate in this unique Project, the Team thanks Cyndi Rader and Roman Tankelevich, as well as for their aid and direction in the Project.

B. Future Directions

FSD has expressed interest in keeping the Team at the company and making them individually paid interns of the company. Every member of the Team appreciates this offer and hopes to see it through shortly after the Summer Session completes. Should the Team be hired at FSD, the Project may be extended to include more aspects of the company, and further *The Link* in the same way.

IV. Glossary

CSM – Colorado School of Mines

Databases – NORMAL and WAREHOUSE

FoxPro – Microsoft developed language and database management system

FSD – Fowler Software Design

LEGACY – The automatically created database in SQL which contains the FoxPro data

NORMAL – The Normal Unified Database

SQL – Structured Query Language; used to gather and sort database information.

The Link – Web Interface used by Fowler Software Design to connect staff to the company. Specifically it is used to document client/staff information.

The Team – Jon Krueger, Branden Salis, Brett Shibao, Philip Suitt

WAREHOUSE – The Warehouse Unified Database

V. References

1. Forta, Ben. *SQL in 10 Minutes*. 2nd. Indianapolis: SAMS, 2001. Print.

VI. Appendix

A. The SQL Databases

A.1 Example SQL Query

First, the Script must determine the database (NORMAL or WAREHOUSE) off of which to base any names or identifiers.

```
USE [FSD_NORMAL_DATABASE]
GO
/***** Object:  StoredProcedure [dbo].[create_invoice_exception_driver]    Script Date:
05/19/2009 10:57:07 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Jon Krueger
-- Create date:    19 May 2009
-- Description:    Create driver table for the Invoice exception report.
-- =====
CREATE PROCEDURE [dbo].[create_invoice_exception_driver]
AS
BEGIN
```

This is where the Stored Procedure is actually defined within the Script.

```
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

--use [FSD_NORMAL_DATABASE]
/*****
- Coding by
- Jon Krueger
- Branden Salis
- Philip Suitt
```

Here is a short overview of the results of this Stored Procedure.

```
-- QUERY OVERVIEW --
This query will show the exceptions between Legacy (FoxPro) and Unified Data per week per
client per tech.
The final table

*****/

-- Disable report manager checking of temporary tables.
SET FMTONLY OFF
```

Due to the use of temporary Tables within the Stored Procedure, it is necessary to make sure none of the temporary Tables exist prior to running or else errors will occur.

```
-- Check for and drop any existing temp tables prior to creating them
-- Unecessary temporary table deletion
if object_id('#Unified_Copy') IS NOT NULL
drop table #Unified_Copy
if object_id('#Legacy_Copy') IS NOT NULL
drop table #Legacy_Copy
if object_id('#No_Null_Copy') IS NOT NULL
drop table #No_Null_Copy
if object_id('#Unified_Null_Values') IS NOT NULL
drop table #Unified_Null_Values
if object_id('#Legacy_Null_Values') IS NOT NULL
drop table #Legacy_Null_Values
```

```

if object_id('#No_Null_Values') IS NOT NULL
drop table #No_Null_Values
if object_id('#unified_temp') IS NOT NULL
drop table #unified_temp
if object_id('#unified_with_sums') IS NOT NULL
drop table #unified_with_sums
if object_id('#legacy_temp') IS NOT NULL
drop table #legacy_temp
if object_id('#Unified_Billing_Exceptions') IS NOT NULL
drop table #Unified_Billing_Exceptions
if object_id('#No_Null_Values_No_Exceptions') IS NOT NULL
drop table #No_Null_Values_No_Exceptions
if object_id('#Final_Exception_Table') IS NOT NULL
drop table #Final_Exception_Table
if object_id('#To_Be_Saved_Exception_Table') IS NOT NULL
drop table #To_Be_Saved_Exception_Table

```

The entire concept of SQL is based off `select * from *` and it is the way that Stored Procedures retrieve data from Tables within the Databases.

```

-- This select statement connects all the Unified data we will be looking at in this
report
select
view_reported_charge.week_id as          week_id,
view_public.code as                    public_code,
view_reported_charge.team_member_charge_amount as charge,
view_staff_member.code as              tech_code
into #unified_temp
from view_public
inner join view_project on view_project.public_id = view_public.public_id
inner join view_requirement_category on view_project.project_id =
view_requirement_category.project_id
inner join view_requirement on view_requirement_category.requirement_category_id =
view_requirement.requirement_category_id
inner join view_reported_charge on view_reported_charge.requirement_id =
view_requirement.requirement_id
inner join view_staff_member on view_staff_member.staff_member_id =
view_reported_charge.staff_member_id

```

In this particular Stored Procedure, the commenting of the code is efficient and useful and should be mostly self-explanatory.

```

-- This select statement copies the Unified data and sums the multiple intervals
-- within each week, within each client, within each tech
select
#unified_temp.week_id as week_id_unified,
#unified_temp.tech_code as tech_code_unified,
#unified_temp.public_code as public_code_unified,
sum(charge) as charge_unified
into #unified_with_sums
from #unified_temp
group by week_id, public_code, tech_code

-- This select statement gathers the Legacy data we will be looking at in this report
select
cast(LEGACY_DATA.dbo.ops_trk_file.trkdat as nvarchar(11)) as date_text, -- Converts
datetime to a human-readable DD MMM YYYY format
-- Converts datetime to an integer. Divides by 7 for week. Subtract 5164 to align with
Unified data week_id
cast(LEGACY_DATA.dbo.ops_trk_file.trkdat as bigint)/7-5164 as week_id,
LEGACY_DATA.dbo.ops_trk_file.trktec as tech_code,
LEGACY_DATA.dbo.ops_trk_file.trkcli as public_code,
sum(LEGACY_DATA.dbo.ops_trk_file.trkvsd) as charge
into #legacy_temp
from LEGACY_DATA.dbo.ops_trk_file
group by trkdat, trkcli, trktec

-- This select statement combines the Legacy and Unified data based on
-- week_id, tech_code, and public_code into one big table with many of NULL entries due
to the outer join

```

```

select
#unified_with_sums.*,
#legacy_temp.*
into #Unified_Billing_Exceptions
from #unified_with_sums
full outer join #legacy_temp on #legacy_temp.week_id = #unified_with_sums.week_id_unified
and #legacy_temp.public_code = #unified_with_sums.public_code_unified
and #legacy_temp.tech_code = #unified_with_sums.tech_code_unified

-- The following selects will contain exceptions in themselves if the data entries were
mistakenly entered.
-- e.g. week_id_unified: 512, tech_code_unified: HODJ, public_id_unified: BENJ -> NULL
logged_hours in reported_charge.logged_hours

-- This select statement copies all entries where the Unified data is NULL
-- from the entire Legacy and Unified table
select
*
into #Unified_Null_Values
from #Unified_Billing_Exceptions
where week_id_unified IS NULL

-- This select statement replaces the NULL values of the Unified values with Legacy data
select
week_id as week_id_unified,
tech_code as tech_code_unified,
public_code as public_code_unified,
'0' as charge_unified,
date_text,
week_id,
tech_code,
public_code,
charge,
'NO UNIFIED DATA' as charge_exception
into #Unified_Copy
from #Unified_Null_Values

-- This select statement copies all entries where the Legacy data is NULL
-- from the entire Legacy and Unified table
select
*
into #Legacy_Null_Values
from #Unified_Billing_Exceptions
where week_id IS NULL

-- This select statement replaces the NULL values of the Legacy values with Unified data
select
week_id_unified,
tech_code_unified,
public_code_unified,
charge_unified,
--cast(cast(((week_id_unified+5164)*7) as datetime) as nvarchar(11)) as date_text,
view_week.week_ending_date as date_text,
week_id_unified as week_id,
tech_code_unified as tech_code,
public_code_unified as public_code,
'0' as charge,
'NO LEGACY DATA' as charge_exception
into #Legacy_Copy
from #Legacy_Null_Values
inner join view_week on view_week.week_id = week_id_unified

-- This select statement copies all entries where the Legacy and Unified data are both
not NULL
select
*
into #No_Null_Values
from #Unified_Billing_Exceptions
where NOT (week_id IS NULL or week_id_unified IS NULL)

-- This select statement removes any non-exceptions from the table

```

```

-- (Non-exception: Any entry where Legacy data matches Unified data exactly)
select
*
into #No_Null_Values_No_Exceptions
from #No_Null_Values
where NOT (charge = charge_unified)

-- This select statement copies the entries from the Legacy-Unified Exception table into
a column-named format
-- equivalent to the two previous Legacy-NULL and Unified-NULL tables (for Union
operation)
select
week_id_unified,
tech_code_unified,
public_code_unified,
charge_unified,
date_text,
week_id,
tech_code,
public_code,
charge,
cast((charge_unified - charge) as varchar) as charge_exception
into #No_Null_Copy
from #No_Null_Values_No_Exceptions

---- These select statements do the 3-part UNION of the Unified-NULL, Legacy-NULL, and
Unified-Legacy Exceptions
---- into one big table
select
week_id_unified,
date_text,
tech_code_unified,
public_code_unified,
charge_unified,
charge,
charge_exception
into #Final_Exception_Table
from #No_Null_Copy
UNION

select
week_id_unified,
date_text,
tech_code_unified,
public_code_unified,
charge_unified,
charge,
charge_exception
from #Unified_Copy
UNION

select
week_id_unified,
date_text,
tech_code_unified,
public_code_unified,
charge_unified,
charge,
charge_exception
from #Legacy_Copy
order by week_id_unified

-- This select statement brings the values for public_name, public_id, staff_name,
staff_id
-- into the final table
select
week_id_unified,
date_text,
tech_code_unified,
public_code_unified,
charge_unified,

```

```

charge,
charge_exception,
view_staff_member.name as                staff_name,
view_public.name as                      public_name,
view_public.public_id as                public_id,
view_staff_member.staff_member_id as tech_id
into #To_Be_Saved_Exception_Table
from #Final_Exception_Table
inner join view_public on view_public.code = public_code_unified
inner join view_staff_member on view_staff_member.code = tech_code_unified

-- This deletes the existing table from the FSD_NORMAL_DATABASE
if exists ( select * from FSD_NORMAL_DATABASE.dbo.foxpro_vs_unified_exceptions)
drop table FSD_NORMAL_DATABASE.dbo.foxpro_vs_unified_exceptions

-- This select statement saves the final table into the FSD_NORMAL_DATABASE
select *
into FSD_NORMAL_DATABASE.dbo.foxpro_vs_unified_exceptions
from dbo.#To_Be_Saved_Exception_Table
order by week_id_unified

```

Below here are some commented-out lines of code which were used to test different methods of obtaining the final Table, or showing currently obtained data for testing purposes.

```

--select * from FSD_NORMAL_DATABASE.dbo.foxpro_vs_unified_exceptions

--CREATE TABLE foxpro_vs_unified_exceptions SELECT * FROM #To_Be_Saved_Exception_Table

--insert into foxpro_vs_unified_exceptions #Final_Exception_Table

```

And like any good programmer would implement, the temporary mess created by this Stored Procedure is cleaned up and erased.

```

-- Unecessary temporary table deletion
if object_id('#Unified_Copy') IS NOT NULL
drop table #Unified_Copy
if object_id('#Legacy_Copy') IS NOT NULL
drop table #Legacy_Copy
if object_id('#No_Null_Copy') IS NOT NULL
drop table #No_Null_Copy
if object_id('#Unified_Null_Values') IS NOT NULL
drop table #Unified_Null_Values
if object_id('#Legacy_Null_Values') IS NOT NULL
drop table #Legacy_Null_Values
if object_id('#No_Null_Values') IS NOT NULL
drop table #No_Null_Values
if object_id('#unified_temp') IS NOT NULL
drop table #unified_temp
if object_id('#unified_with_sums') IS NOT NULL
drop table #unified_with_sums
if object_id('#legacy_temp') IS NOT NULL
drop table #legacy_temp
if object_id('#Unified_Billing_Exceptions') IS NOT NULL
drop table #Unified_Billing_Exceptions
if object_id('#No_Null_Values_No_Exceptions') IS NOT NULL
drop table #No_Null_Values_No_Exceptions
if object_id('#Final_Exception_Table') IS NOT NULL
drop table #Final_Exception_Table
if object_id('#To_Be_Saved_Exception_Table') IS NOT NULL
drop table #To_Be_Saved_Exception_Table

```

All Stored Procedures are finished by the single command END
END

B. The Report Manager

B.1 Example Report of the Project



14 June 2009

List of Requirements by Category

FSD

Automation Internship

Report Rate Per Hour: \$0.00

		Benchmark Budget	Actual Cost	Used Hours	Remaining Hours	Estimated Hours	Estimated Cost	Percent Complete
01	Development							
1.1	Activity Reports [1]	0.00	0.00	0.00	0.00	0.00	0.00	%
1.3	The Link Interface [7]	0.00	0.00	0.00	0.00	0.00	0.00	%
1.2	Miscellaneous [6]	0.00	0.00	0.00	0.00	0.00	0.00	%
	Totals:	0.00	0.00	0.00	0.00	0.00	0.00	%
02	Company Introduction							
2.1	Meeting Members [3]	0.00	0.00	0.00	0.00	0.00	0.00	%
2.2	Company Information [4]	0.00	0.00	0.00	0.00	0.00	0.00	%
	Totals:	0.00	0.00	0.00	0.00	0.00	0.00	%
03	Testing							
3.1	User Interface [2]	0.00	0.00	0.00	0.00	0.00	0.00	%
3.2	Activity Reports [5]	0.00	0.00	0.00	0.00	0.00	0.00	%
	Totals:	0.00	0.00	0.00	0.00	0.00	0.00	%
04	Documents							
4.1	CSM - Field Session [8]	0.00	0.00	0.00	0.00	0.00	0.00	%
	Totals:	0.00	0.00	0.00	0.00	0.00	0.00	%
05	Database Management							
5.1	Tons o SQL [9]	0.00	0.00	0.00	0.00	0.00	0.00	%

C. *The Links*

C.1 *The Link* User Guide

The original Guide was created by Becca Hubis from FSD, but was appended for the recent updates to *The Link* by the Project.

Fowler UDS System Guide

System Administrator Roles

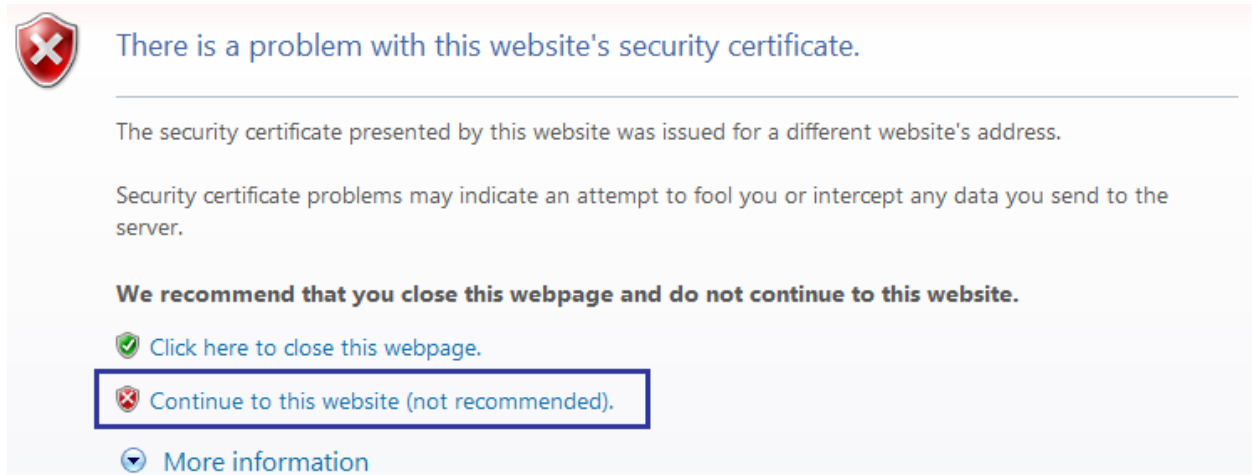
Becca Hubis
12 May 2009
Edited: Jon Krueger
11 Jun 2009

Table of Contents

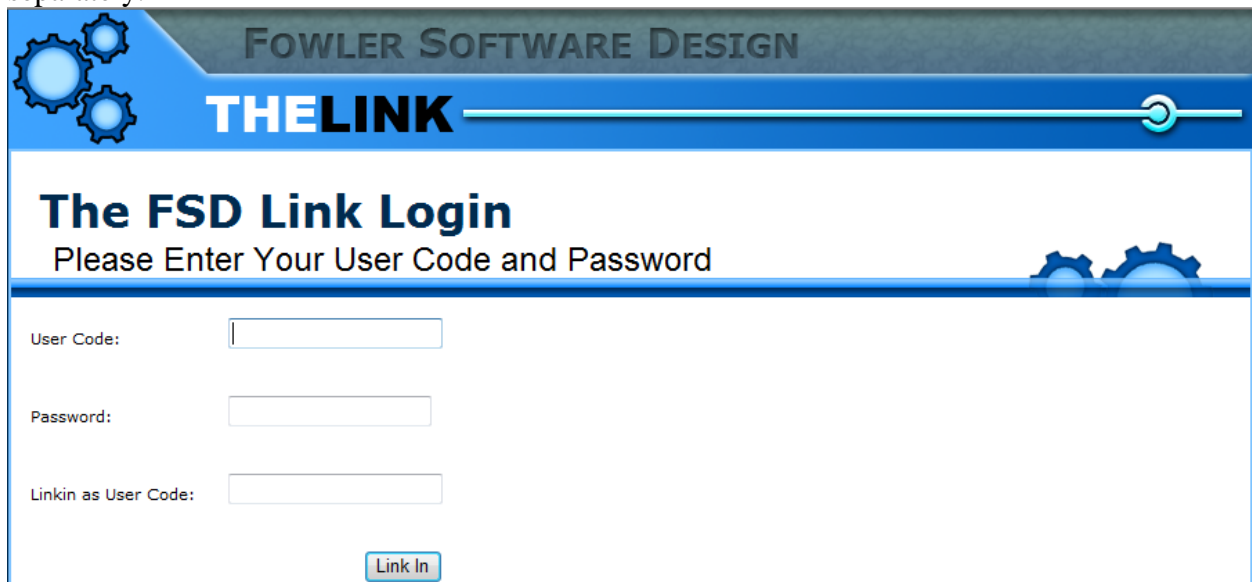
<u>Logging into the Link</u>	25
<u>User Main Menu</u>	27
<u>Creating A New Staff Member/ User</u>	28
<u>Creating a New Client</u>	8
<u>Creating a New Project</u>	32
<u>Recording Your Time in The Link</u>	36
<u>Invoicing Time</u>	38
<u>Adding Other Charges to a Customer Invoice</u>	39
<u>Credit Memos</u>	17
<u>Override Percents</u>	18
<u>Logging Off of The Link</u>	20

Logging into the Link

To Access the Link, first open an internet browser, such as internet explorer. Enter [://uds.fowersoftware.com](http://uds.fowersoftware.com) into the browser address bar and go to the site. At this point, you may run into a certificate error, such as the one below:



If you encounter this, chose “Continue to this website (not recommended).” At this point, you will be directed to the Login Page, as depicted below. You will be prompted to enter your User Code and Password. Your User Code is the first three letters of your last name followed by your first initial. For example, John Smith would login as SMIJ. Your password will be sent to you separately.

A screenshot of the login page for Fowler Software Design. The header features the company name "FOWLER SOFTWARE DESIGN" and "THELINK" in large blue letters. Below the header, the page title is "The FSD Link Login" with the instruction "Please Enter Your User Code and Password". There are three input fields: "User Code:", "Password:", and "Linkin as User Code:". A blue "Link In" button is positioned below the fields. The page has a blue and grey color scheme with gear icons.

Once you have entered your User Code and Password, click the “Link In” button. You will be directed to your User Main Menu. You are now logged in.

User Main Menu

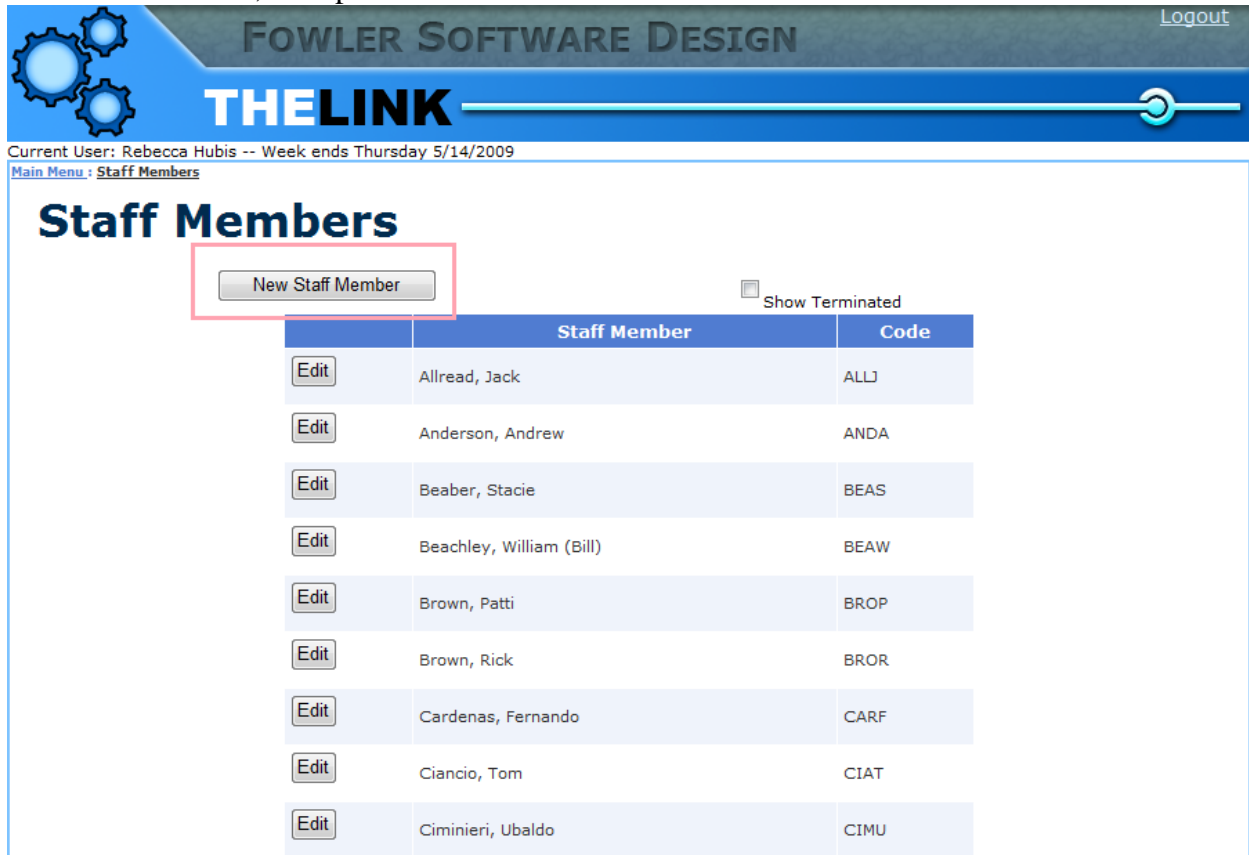
Below is a screen capture of the User Main menu. You may or may not have all of the same options depending on your accessibility.

The screenshot shows the 'User Main Menu' interface. At the top, there is a blue header with the text 'FOWLER SOFTWARE DESIGN' and a 'Logout' link. Below the header is a blue bar with the logo 'THELINK' and a circular refresh icon. A status bar below the header displays 'Current User: Rebecca Hubis -- Week ends Thursday 5/14/2009'. The main content area is titled 'Main Menu' and contains nine blue buttons arranged in a 3x3 grid. Each button has a gear icon on the left and a text label. The buttons are: TIME TRACKER, ACTIVITIES, STAFF MEMBERS, PROJECTS, CLIENTS, PROJECT PROPOSAL, OPERATOR, INVOICE TIME, and OTHER CHARGES.

TIME TRACKER	ACTIVITIES	STAFF MEMBERS
PROJECTS	CLIENTS	PROJECT PROPOSAL
OPERATOR	INVOICE TIME	OTHER CHARGES

Creating A New Staff Member/ User

From the Main Menu, select the “Staff Members” option. You will be directed to the “Staff Members” page, with a list of all Current and Terminated users. There is also the option to add a New Staff Member, as depicted below:



Current User: Rebecca Hubis -- Week ends Thursday 5/14/2009
Main Menu : [Staff Members](#)

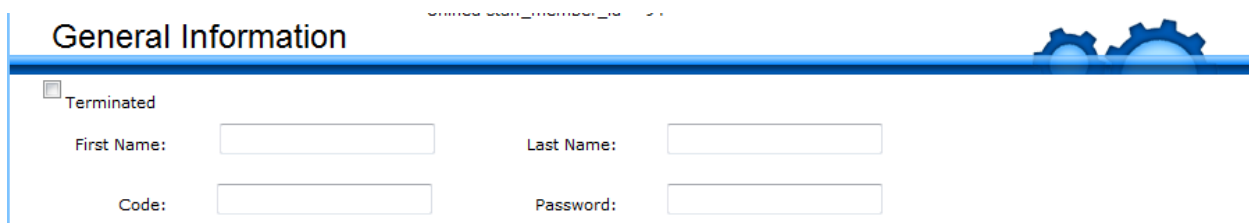
Staff Members

Show Terminated

	Staff Member	Code
<input type="button" value="Edit"/>	Allread, Jack	ALLJ
<input type="button" value="Edit"/>	Anderson, Andrew	ANDA
<input type="button" value="Edit"/>	Beaber, Stacie	BEAS
<input type="button" value="Edit"/>	Beachley, William (Bill)	BEAW
<input type="button" value="Edit"/>	Brown, Patti	BROP
<input type="button" value="Edit"/>	Brown, Rick	BROR
<input type="button" value="Edit"/>	Cardenas, Fernando	CARF
<input type="button" value="Edit"/>	Ciancio, Tom	CIAT
<input type="button" value="Edit"/>	Ciminieri, Ubaldo	CIMU

Select the “New Staff Member” option, and you will be taken to the “New Staff Member” page where you will be prompted to enter 4 categories of information: General Information, Billable Staff Member, Staff Member Type, and Permissions.

General Information



General Information

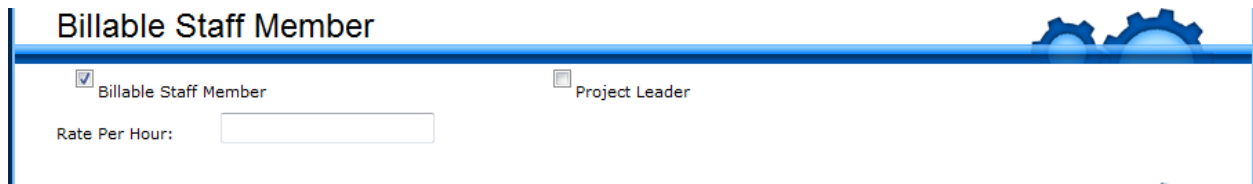
Terminated

First Name: Last Name:

Code: Password:

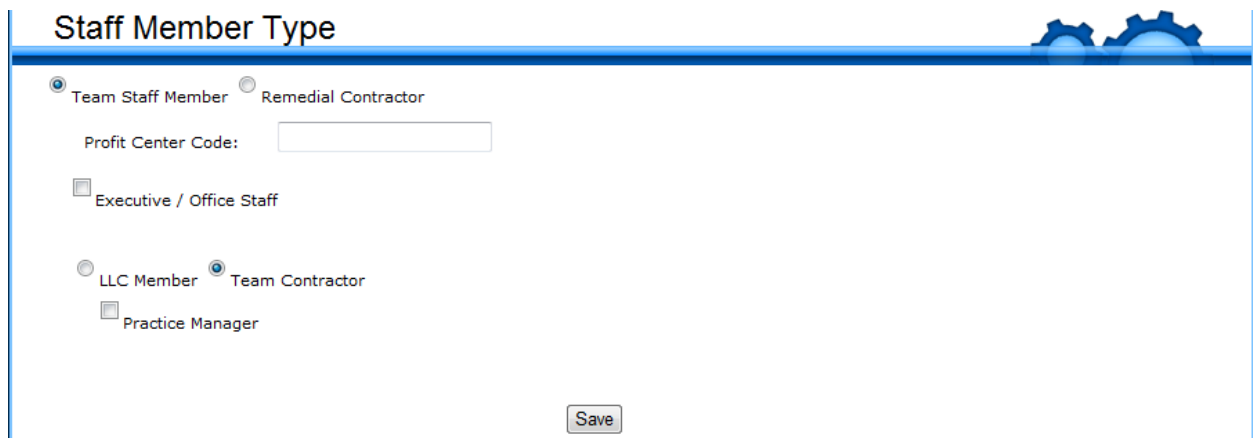
Here you will be prompted to enter the Users Name, Code, and Password. Code is the 4-letter User Code used to Login.

Billable Staff Member



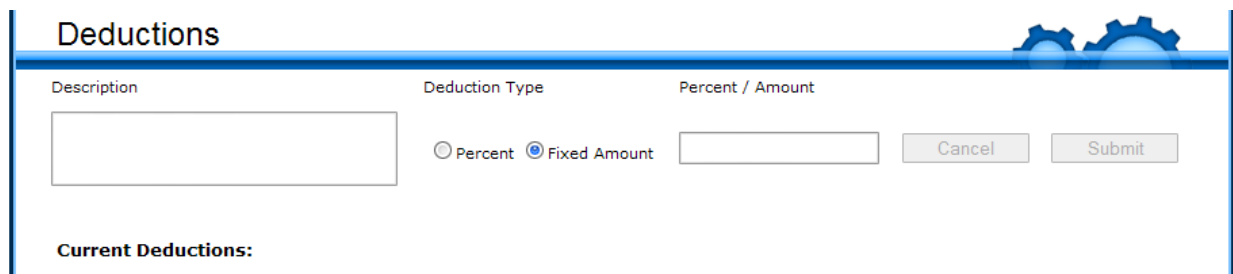
If the User is a Billable Staff Member – the client will be build for work preformed – select the Billable Staff Member Box. This will bring up the screen above, and allow you to select whether the person is a Project Leader and the Billable Rate per Hour.

Staff Member Type



Next, enter information regarding Staff Member Type. The Profit Center Code is again the 4-character User Code (first 3 letters of last name followed by first initial). Most likely, “Team Staff” member will be selected, and if the user is not a programmer or tester, select the “Executive/Office Staff” option and enter the required information. Also, unless otherwise noted, the user will be a Team Contractor. Make sure to save all three sections by selecting save.

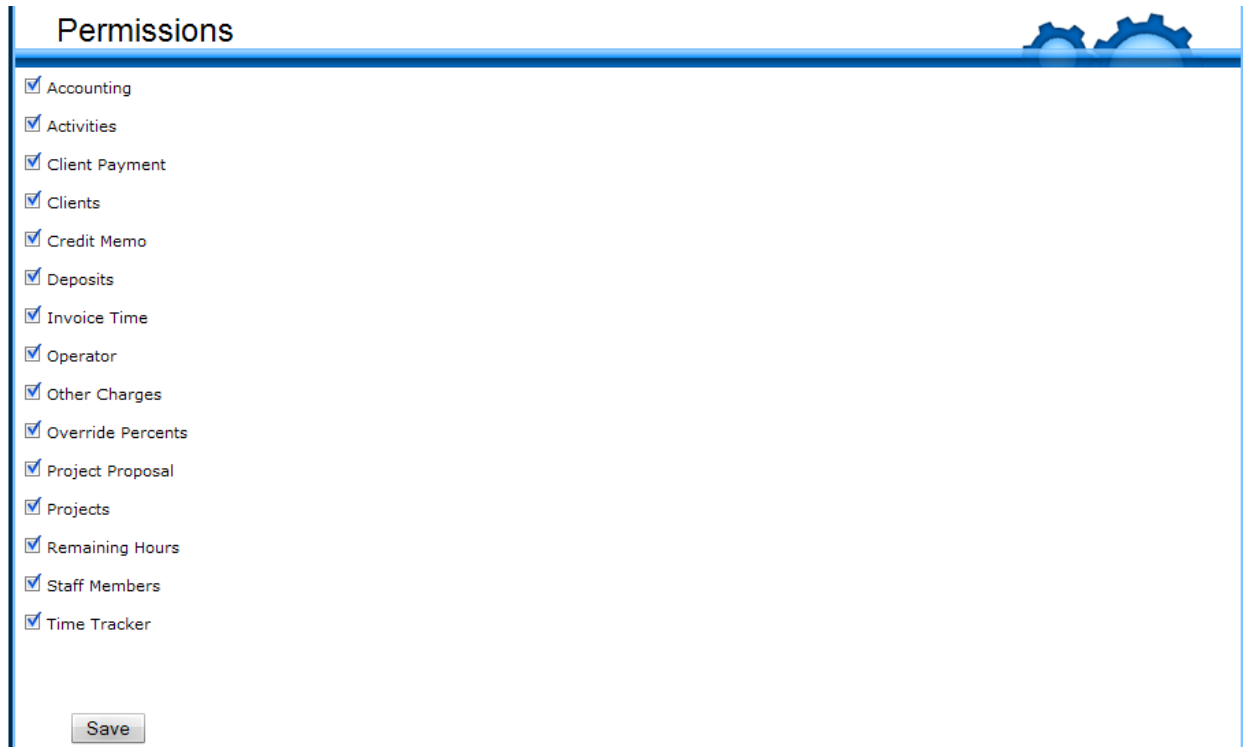
Deductions



This is the section where Deductions can be added for individual staff members. Percents are entered as integers without decimals, and amounts can be any number less than 10000.00. Both

percent and amount values must be greater than 0. The Percent/Amount box value is specified by the radio button. Any current deductions will be listed below with their description and percentage/amount. Next to all current deductions are “Deactivate” buttons to deactivate the particular deduction for the current and future weeks.

Permissions



Permissions

- Accounting
- Activities
- Client Payment
- Clients
- Credit Memo
- Deposits
- Invoice Time
- Operator
- Other Charges
- Override Percents
- Project Proposal
- Projects
- Remaining Hours
- Staff Members
- Time Tracker


Save

To finish creating the customer, add the appropriate Permissions. Everyone should have access to Time Tracker and Activities, in order to maintain hours on the Link. Be sure to press save after selecting permissions.

The New User is now created. To Edit the User, return to the “Staff Members” page and select the “Edit” option associated with the User.

Creating a New Client

In order to create a New Client, select the “Clients” option from the User Main Menu. This will direct you to the “Clients” page, which is very similar to the “Staff Members” page, just lists Clients instead. Select the option to add a new client, and you will be directed to the “Add New Client” page.

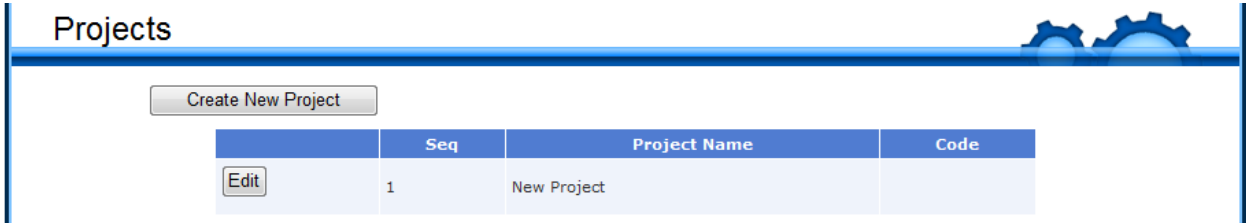


The screenshot shows the 'Edit Client' page in the THELINK system. The page header includes 'FOWLER SOFTWARE DESIGN' and 'THELINK'. The current user is identified as 'Rebecca Hubis' and the date is 'Thursday 5/14/2009'. The main heading is 'Edit Client' with a sub-heading 'Client Information'. The form contains three input fields: 'Client Name' with the value 'A New Client', 'Client Code' (empty), and 'Closed' (checkbox). A 'Save Client Information' button is located at the bottom right of the form area.

Here you will be prompted to enter a Client Name and Code. The code is a 4-character code that will be used as a sort of Acronym for the client name. Click the “Save Client Information” button to save the client. The Client has now been successfully created.

Creating a New Project

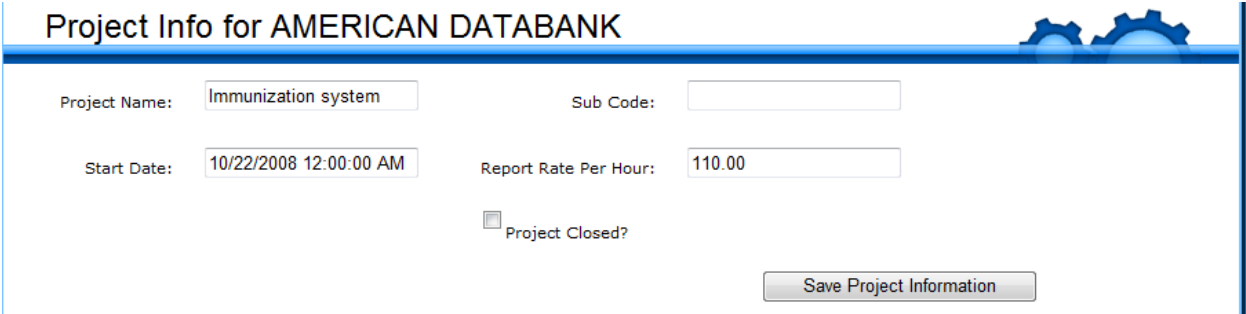
To create a New Project, first select the “Clients” link. From there you will be taken to a list of the clients. Select the “Edit” button associated with client you wish to add the Project to. From there you will be taken to the Edit Client page. Scroll to the “Projects Section”:



This section will list all the previous and current projects for that Client. Select the option to “Create New Project”. You will see a “New Project” appear on the list of Projects. Select the Edit button to the left of the New Project.

This will bring you to the Edit Project Page where you will be able to enter client information into 4 sections: Project Info, Requirement Categories, and Team Info.

Project Info



In this section, you are prompted to Name the project, add a Sub Code, Start Date, and Report Rate per Hour. The Sub Code is optional and the Report Rate per Hour would be information to get from the Project Lead. Remember to save the Project Information.

Requirement Categories

Requirement Categories

(Click the requirement category name to edit.)

- ▶ 1 [Design](#)
- ▶ 2 [Development](#)
- ▶ 3 [Testing](#)
- ▶ 4 [Change Requests](#)
- ▶ 5 [Onsite Request](#)

Above is the “Requirement Categories” for an active project. As you can see Requirement Activities are the actions taken on a project by a team member. You can create a New Requirement Category by selecting the button. This will add a default Category to the end of the list. By clicking the new category, you will be taken to the “Edit Requirement Category” page.

Edit Requirement Category

Edit Requirement Category

Requirement Category Info

Category Name:

Sequence: 2

List of Requirements

Number	Name	Benchmark Budget	Remaining Hours	Estimated Other Cost	
4	<input type="text" value="Database"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
5	<input type="text" value="Student Web Pages and Graph"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
6	<input type="text" value="School Admin Web Pages"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
7	<input type="text" value="System Admin Web Pages"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
8	<input type="text" value="Reminder System"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
9	<input type="text" value="Custom Controls, Tools and Mi"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
10	<input type="text" value="Testing"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
11	<input type="text" value="Project Management and Admini"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>
13	<input type="text" value="Compliance Rules"/>	<input type="text" value="0.00"/>	<input type="text" value="0.000"/>	<input type="text" value="0.00"/>	<input type="button" value="Delete"/>

Each Category has a List of Requirements. The List of Requirements allows the client to see a more specific description. So, instead of just reporting Development, you can report Development – Database.

A project can have as many categories and requirements as necessary.

Team Info

Team Info

Project Leader: none

Team Members:

Name	Code		
Bob Read	REAB	<input type="button" value="Make Project Leader"/>	<input type="button" value="Remove From Project"/>
Mitch Dusina	DUSM	<input type="button" value="Make Project Leader"/>	<input type="button" value="Remove From Project"/>
Rebecca Hubis	HUBR	<input type="button" value="Make Project Leader"/>	<input type="button" value="Remove From Project"/>
Tom Ciancio	CIAT	<input type="button" value="Make Project Leader"/>	<input type="button" value="Remove From Project"/>

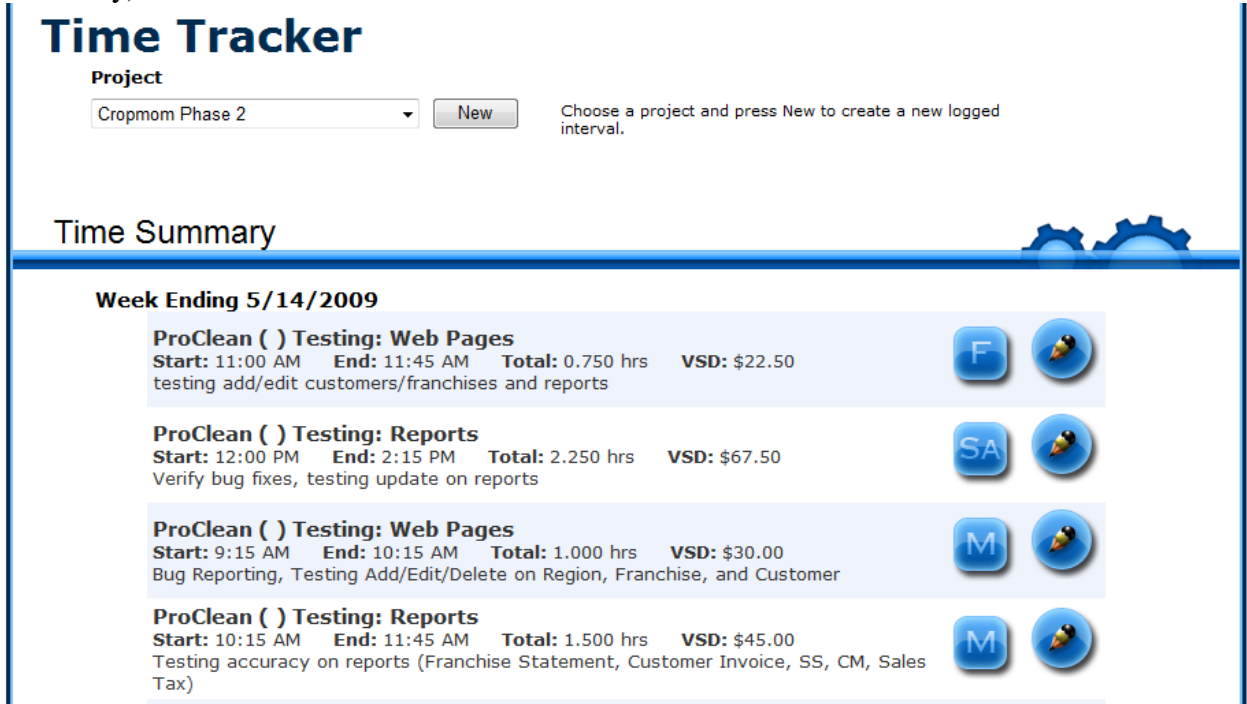
Search for Team Members:

Name: Code:

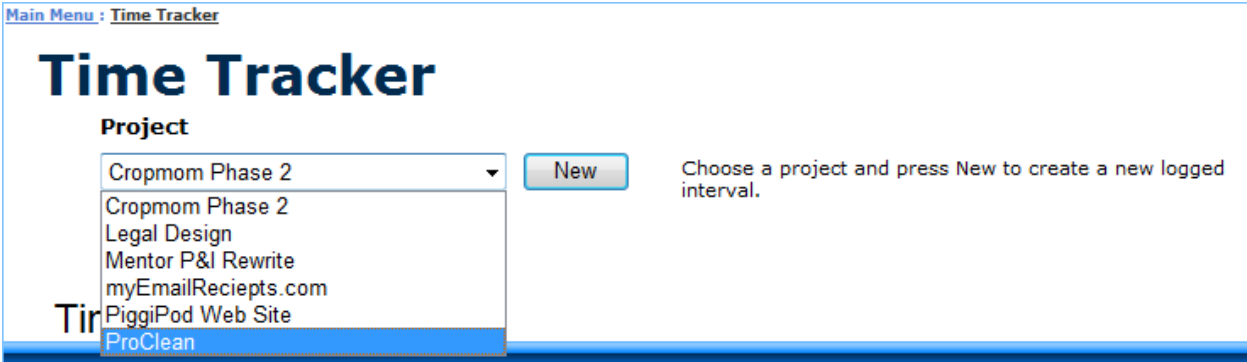
The Team Info information allows the team members to be able to access the project in their activities link. If a user is not a Team Member, they cannot record their hours to that project. To add team members, simply search of the person, and select the option to add them to the project. Once you have entered the Project Information, Requirement Categories (with appropriate requirements), and Team Members, the project has been created. Project can be accessed through the Client pages, or also by selecting the “Projects” option from the User Main Menu. This will take you to a list of projects.

Recording Your Time in The Link

To record your time, select the “Time Tracker” option from the Main Menu page. This will bring you to the Time Tracker page where you are able to enter your hours, as well as see a list of the hours you have logged for the current fiscal week (Time Summary):



To enter time in to the link, start by selecting the project from the drop down menu:



Once you have the correct project, press the “New” button. You will then be prompted to select your activity from the Activity drop down list. Basically, what did you do during this logged interval? The Rate will automatically populate based upon the information entered when creating the user. Simply select the day and set the work time frame. Then, add a more detail description of what you do. This description is printed on to the receipt given to the Client at the end of each billing period. See image below:

Time Tracker

Project

ProClean

New

Now creating a **new** logged interval. Press Submit to save. You can also change which project this new logged interval will be logged under.

Activity

Testing Performance Tuning

Rate

30

Day

Tuesday

Start

900

End

AM

500

PM

Description

This logged interval is free work.

Submit

Cancel

After you have entered everything, hit the “Submit” button and the hours just logged will show in the appropriate chronological order on User’s Time Summary.

If you cannot find the appropriate activity, you may need to Add New Activities.

Adding a New Activity

From the Main Menu, select the “Activities” button. There are 3 sections on the Activities page: Requirement Activities, Non Requirement Activities, and User’s List of Activities. The Non-Requirement Activities are activities for which you are not billing the client. The User’s List of Activities is just a list of all of the Activities the User has added.

Requirement Activities

Requirement Activities

Project: Braun Production Unit

Category: Project Management

Requirement: Project Management

Create New Requirement Activity

This is where you are able to add additional Activities to show in the Time Tracker. Simply select the Project, Category and Requirement most suited for you work and click the “Create New Requirement Activity” button. You have just created a new activity to log hours to in the time tracker.

Invoicing Time

After entering hours, you are able to see and approve all of the hours logged for a particular project for the week. This is done through the “Invoice Time” option on the Main Menu. This brings you to the “Invoice Time” page. Here you can see who has recorded what for each project and approve time entered to invoice the Client. To see other projects, press the “Previous” or “Next” buttons and different projects will be displayed.

Invoice Time

Client Name: Fullerton Investors
Project Name: ProClean
Amount to Invoice: \$ 0

Logged Intervals

All Logged Intervals

<input type="checkbox"/>	Elise Harpel	Total: \$0
<input type="checkbox"/>	5/8/2009 12:30:00 PM FULL Testing: Web Pages (3.500 hours) Tested Fiscal Summary Spreadsheet	\$105.00
<input type="checkbox"/>	Fernando Cardenas	Total: \$0
<input type="checkbox"/>	5/8/2009 9:00:00 AM FULL System Modeling and Generation: Application Generation (1.500 hours) Made some UI changes to support transfer of franchise	\$187.50
<input type="checkbox"/>	5/8/2009 10:30:00 PM FULL Customer Pages: Create Customer Page (0.500 hours) Fixing a small bug with reporting	\$62.50
<input type="checkbox"/>	5/11/2009 9:00:00 AM FULL Customer Pages: Create Customer Page (1.000 hours) Added in all the states and made some minor adjustments to the UI for transferring a franchise	\$125.00
<input type="checkbox"/>	Mitch Dusina	Total: \$0
<input type="checkbox"/>	5/11/2009 12:30:00 PM FULL Franchise Pages: Create Franchise Page (0.750 hours) Bug 33	\$56.25

Adding Other Charges to a Customer Invoice

Occasionally, we will need to Invoice a Client for an additional charge. To do this, simply select “Other Charges” from the Main Menu. This will bring you to a page where you can enter the required information and save:

Other Charges

Week Ending Date: 5/14/2009

Client:

Project:

Category:

Requirement:

Description:

Amount:

Submit

Cancel

Credit Memo

The Credit Memo page is used to make corrections to invoice amounts. In the case of an overcharged or and undercharged project for the current week, the credit memo allows for easy changes to the total invoice amount for that project.

Credit Memo

Week Ending Date: 6/11/2009

Client

Project

Credit Amount

Note: This credit memo will only update for the current week, so no changes to previous week's invoices can be made once the week is closed. However, credit memos which are added on projects which do not have any VSD for the current week will still print an invoice for that project.

Selecting the **Client** from the drop-down will reveal all of that client's projects in the **Project** drop-down. Simply select the correct project and put the value of credit in the **Credit Amount** box. Once selections have been made, the Submit button will activate, and pressing it will add the amount onto a new invoice for that project, for that client.

Note: Credit amounts can be negative or positive. Remember that the amount is ADDED to that project on a new invoice, so positive values will result in charging the client more, and negative values will result in compensation to the client for past errors on an invoice.

Override Percents

A new addition to The Link is the ability to set override percentages at different frequencies for various groups. These overrides will replace the standard percentages as also defined on this page. As you can see from the screenshot below, when no standard percents are changed, and no overrides are set, the respective Submit buttons will remain inactive. Pressing cancel in either section should clear the non-default values in that section and make no database changes.

Override Percents

System Parameters

Standard Tech Percent

Standard Project Leader Percent

Standard Practice Manager Percent

Override Percents

Billable Staff Member - Tech Percent	<input type="text"/>	Staff Member	<input type="text" value="select staff member"/>
Team Member - Tech Percent	<input type="text"/>	Project	<input type="text" value="select project"/>
Project Leader - Project Leader Percent	<input type="text"/>	Project	<input type="text" value="select project"/>
Project - Project Leader Percent	<input type="text"/>	Client	<input type="text" value="select client"/>
Practice Manager - Practice Manager Percent	<input type="text"/>		
Client - Practice Manager Percent	<input type="text"/>		

Quantity Percents

Tech Quantity Percent

Project Leader Quantity Percent

Practice Manager Quantity Percent

System Parameters

These are the standard percents which are applied to all situations where no override percents in that category are supplied.

Tech percent will be the default percentage of tech VSD that the tech will receive between both invoice completion (quantity), and full invoice payment by the client (quality). **Project Leader percent** is the default percentage of total project VSD for each week that the project leader of said project will receive between both quantity and quality payments. **Practice Manager Percent** is the default percentage of total VSD for the given client for each week that the client's practice manager will receive between both quantity and quality payments.

Override Percents

Override percents work in a priority basis when they are determined. These overrides and the standard percents will be used to obtain the actual percent within team member for any given percent type. They are as follows

Team Member: This is a tech-specific percent and is the first priority for a tech percent override. This is specific to the project drop-down on the right of this box, and the tech specified at the top of the section.

Billable Staff Member: This is a tech-specific percent and is the second priority for a tech percent override. This is specific to the tech listed at the top of the section.

Project: This is a project-leader-specific percent and is the first priority for a project leader override. This is specific to the project drop-down to the right of this box only.

Project Leader: This is a project-leader-specific percent and is the second priority for a project leader override. This is specific to a tech who is also a project leader for any given project specified by the tech drop-down at the top of the section. *(Note: If the selected tech is never a project leader, setting this value will have no effect.)*

Client: This is a practice-manager-specific percent and is the first priority for a practice manager override. This is specific to a client and therefore the practice manager of that client, regardless of the tech drop-down box at the top of the section.

Practice Manager: This is a practice-manager-specific percent and is the second priority for a practice manager override. This is specific to the tech who is a practice manager for any clients, and is specified by the tech drop-down box at the top of the section. *(Note: If the selected tech is never a practice manager, setting this value will have no effect.)*

Quantity Percents

These percents represent the percent of compensation received at invoice time. The remaining amount to sum with "quantity" percent to 100% is considered "quality" percent and is paid to the staff member at the time at which the invoice is fully paid off by the client.

Tech Quantity Percent: This is the quantity percent for every tech.

Project Leader Quantity Percent: This is the quantity percent for every project leader.

Practice Manager Quantity Percent: This is the quantity percent for every practice manager.

(Note: There are no overrides for quantity percents. The amounts specified in these boxes are specific for the current week close out and for all members of each category.)

Logging Off of The Link

To log off of the link, simply select the “Logout” link at the top right of every page:

