Directory Web Site Management System

By

Jon Zielinski

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

Directory Web Site Management System

By

Jon Zielinski

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

Jon Zielinski	Date
Professor Russell McMahon, Faculty Advisor	Date
Patrick C. Kumpf, Department Head	Date

Acknowledgements

I would like to thank Andrew Wash for having faith in me to provide him with a service and product that will help represent his company to his clients. I would like to thank Professor McMahon for all the assistance he has given me. I would also like to thank Dwain Gober, Chas Godbey, and Randy Theis for trying to break my project during the testing process. I would finally like to thank all my family and friends, especially my parents and my brother Don for supporting me during my college years.

Table of Contents

Listing of Figures

Abstract

Valley Solutions, Inc. is a small business located in West Chester, Ohio. They provide advertising for other companies in the form of directory Web sites targeting particular service customers are looking for. The current backend they are using for their Web sites is using outdated technology, and requires each Web site to run its own instance of the backend application. Valley Solutions, Inc. has requested that a new backend application be developed. They requested that this one application be flexible enough to accommodate many Web sites that use a similar data schema. This will allow Valley Solutions a centralized point to administer the sites using the backend. The application will be created using ASP.NET using C# and SQL Server. The backend application will write directory Web pages based on an existing template provided by an administrator. Also, clients will be able to directly manage their company and listing information. Finally, the backend application will also produce invoices for a particular client monthly, and E-mail the same client when the invoice information is ready to be viewed.

1. Statement of Problem

Valley Solutions, Inc. is a small business located in Cincinnati, Ohio. This company has many Web sites that provide services for clients from different categories of business. It has many Web sites that show information in a similar fashion, but have different backend applications running them. Each of the company's Web sites, which are located on the same server, use separate databases and different background applications. Most of the backend processes run using a dynamic link library (DLL) written in Visual Basic 6.0. Each DLL has been modified slightly to access the different databases, though the format of the data remains consistent in each database. Most of the databases are on SQL Server 2000, but a few are in Microsoft Access format. The current process also has different means of client management. Many of the sites currently run by Valley Solutions. Do not allow the client to modify directly his/her company information located in the databases. This includes new customers that wish to use Valley Solutions services. Most data, which is entered or modified in the database, are a result of copying and pasting information from an e-mail, which is sent from a form located on a particular Web site. This could delay when the information is added or changed on a particular Web site. This is a time consuming process.

2. Description of the Solution

The Directory Web Site Management System will initially be used to manage one site, but will have the flexibility to be used with multiple sites. This system will streamline the customer service process for clients. The system will allow the client to modify or add information related to his/her account directly to the database. The index site management system will also have the ability to create the site structure (i.e., folders and files) and the content of the desired pages dynamically, including the main page. The pages will be written in HTML, but will be dynamic in the sense that the pages will be recreated on a regular basis to update newer content. The system will also provide tools useful to the day-to-day operation of the site. These tools include automated e-mail for service renewal, reports, and site page management.

2.1 User Profile

There are three levels of application end-users. The three levels include administrators, clients and users.

2.1.1 Administrators

These are the employees of Valley Solution, Inc. Administrators will have the ability to approve, reject, add, modify, or delete listings from any client. Also, administrators have the ability to modify client account settings. And have the ability to create a new directory using his/her graphic user interface, and the ability to modify settings of currently directories.

Administrators should have basic computer knowledge. They will be able to navigate the applications tools and use them after a small amount of training.

2.1.2 Clients

Clients are end-users who work for companies outside of Valley Solutions. These companies sign up for the service that Valley Solutions provides which is directory listing. Clients will have the ability to modify his/her account information, as well as to add and modify listings. They will not have the ability to delete listings since they will need to agree for a contract to a specific time his/her listing will appear. Once a listing is added or modified, those listings are sent to the administrator end-users for approval or denial.

Clients have unknown computer knowledge, but should know how to navigate a Web Site, and be able to enter their company information into a Web based form and submit it. Clients will have the option of calling Valley Solutions. For assistance if they can not navigate the application on his/her own.

2.1.3 Users

Users are individuals that browse any of Valley Solutions directory sites, though they are not paying clients. Users are searching the listings of clients listed on Valley Solutions sites. They are interested in a service that one of Valley Solutions clients provides. They will only interact with the Web Site features that are located outside of client and administrator login areas.

Users have unknown computer knowledge, but should know how to navigate a Web page. Limited support will be offered to Users, including answering any questions regarding Valley Solutions and/or its Web Sites. Questions from a user about a particular client will be sent to the client for a response.

2.2 Design Protocols

2.2.1. Organizational Scheme

The overall organizational scheme is based on the design of a single application that will be used by different Web sites. The application will detect what URL is being referenced by the client and show the appropriate information for that site.

The application will be organized into 2 separate areas, administrator and client. The administration area will be organized into the following areas:

- **Client Administration Utilities:** this section allows administrators to add, modify, or change information for a client. This information includes company information and listing information. The administrator is also able to approve or reject pending listings as well as assign existing listings to a Web page.
- **Site Administration Utilities:** this section allows administrators to add, modify, or change site information. This includes adding the URL of sites that will use the application. Assigning categories used on that particular site and also managing category descriptions and keywords that will aid the site with search engine recognition. Also, the application will have the ability to create a new header and footer for pages that are not created using the application. Finally this section will also allow the end-user to recreate the main page with updated information, create the directory pages, and individual client pages including company listings and informational pages.
- **Report Utilities:** this section will allow the administrative end-user to view in a useful way information regarding revenue generated, number of listings active, soon to expire listings, etc.

The client area will be separated into the following sections:

- **Add New Company:** this section will allow clients to add new company information.

- **View Company Information:** this section will allow clients to view, or modify their company information. This section will also allow clients to add or modify listing information. If there are approved listings, the client will be able to add the listing to a particular page on the directory.
- **View Pending or Rejected Listings:** this section will allow the end-user to modify information on a listing that has not yet been approved.

2.2.2. Interface Design/Navigation

The interface is a simple consisting of links, and forms. The Web site is being optimized for an end-user resolution of 1024 x 768. The navigation buttons used in forms are standard submit buttons created using ASP.NET.

2.2.3. Icons/Graphical Symbols

Currently, no icons or graphics are included in the design of the application. However, during the testing process of the application, it is expected that a header and footer will be added to the applications HTML page for the appropriate page. This will be accomplished using either using Internet Information Services or ASP server side includes.

2.2.4. Color Scheme

Currently no customizations are being made to the application. However, once the application is customized for a particular site, the administrator will have the ability to change the color scheme to fit the Web site using Cascading Style Sheets (CSS).

3. Deliverables

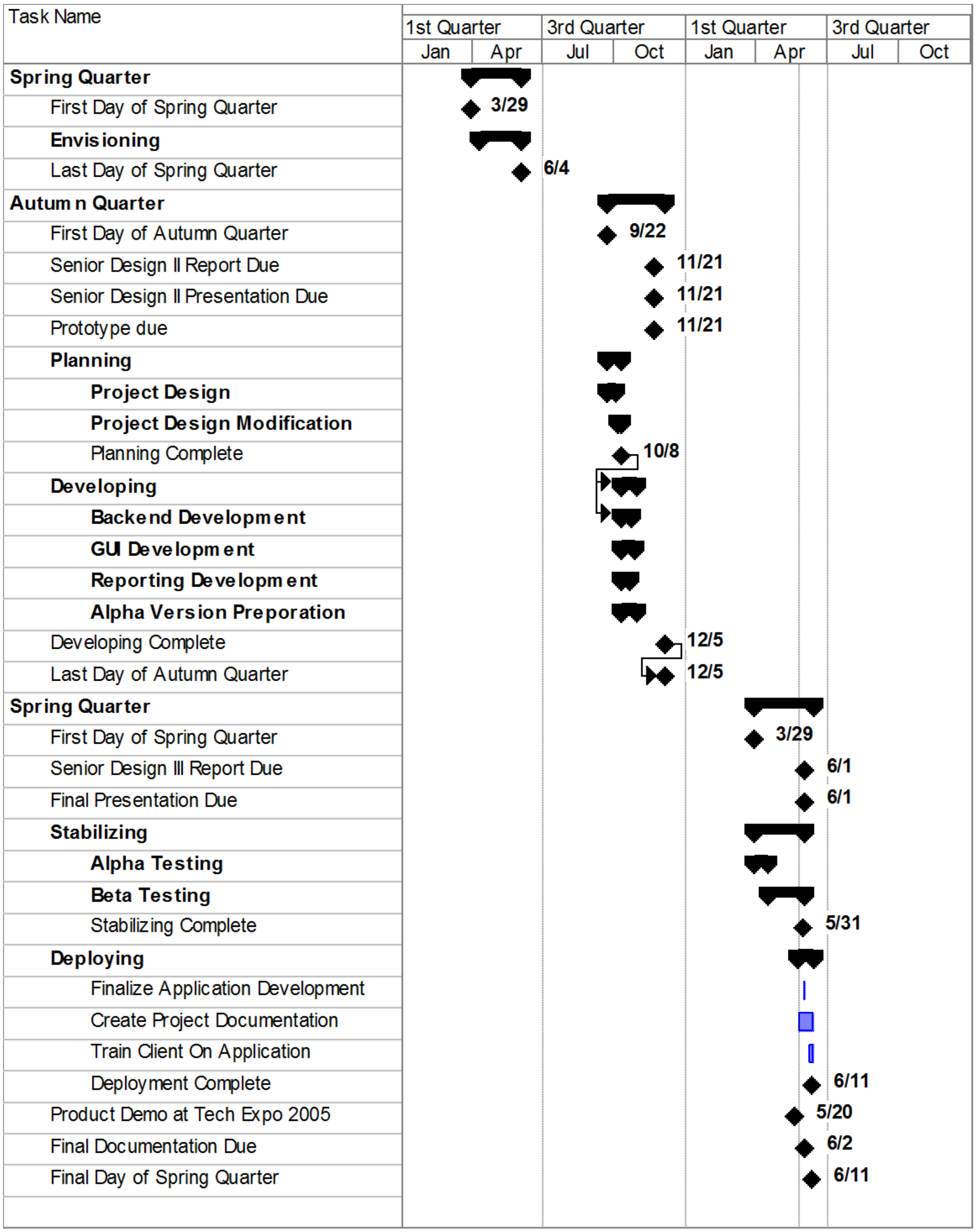
1. A Web Site Management System that allows administrators and clients to modify company and listing information.
2. The user interface will be written in HTML and ASP.NET using C#.
3. The database will be created and managed using SQL Server.
4. New clients of the Directory Web Site Management System will be able to create a profile.
5. Administrators will be able to add/remove/delete directories as well as the directories category information.
6. The Web Site will have the ability to create all directory pages at the administrator's discretion.

Administrators will be able to pull up a report detailing which listings are active for a particular date range, as well as the amount due for listing. Administrators will then be able to generate an E-mail to each company detailing the amount due for that month.

4. Design and Development

The following sections provide an overview of the project's timeline, and budget.

4.1. Timeline



4.1.1. Senior Design I Accomplishments

During Senior Design I, I accomplished the following:

- Analyzed current system used by Valley Solutions, Inc.
- Researched Valley Solutions, Inc. needs.
- Compared available products that could fulfill needs of Valley Solutions, Inc.
- Developed prototype of user interface, site navigation, and sample features.
- Developed database.
- Developed proposal and oral presentation.

During the research portion of Senior Design I, I spent time looking for products that meet the needs of Valley Solutions, Inc. I found a few products that were portal builders. These products were similar to the needs of Valley Solutions; however it was missing features that Valley Solutions was interested in, and the cost was out side of their price range. I designed the database and site navigation based on Valley Solutions existing database and site navigation. I then expanded on those templates basing them on a backend that would be used by multiple sites. I also simplified some database tables that required redundant data to be entered attempting to streamline the backend process as much as possible.

4.1.2. Senior Design II Accomplishments

During Senior Design II, I accomplished the following:

- Created functional prototype using completed features.
- Previewed completed features during demonstration of prototype.
- Prepared Design Freeze documentation and oral presentation.

During this portion of the Senior Design sequence, I had to learn how to program in ASP.NET using C#. I had a basic understanding of how to program in .NET from my C# classes, and previous experience coding in ASP 3.0. I had to learn how they differed, and how to accomplish what I used to know under ASP.NET. During this time, as different components of the Web site were completed, I would test the completed component to make sure the desired output would be obtained.

4.1.3. Senior Design III Accomplishments

During Senior Design II, I accomplished the following:

- Completed Web site programming.
- Tested site.
- Added error checking as needed.
- Modified project as needed.
- Completed Final Documentation for the project.
- Presented the final project at Tech Expo 2005.

During this portion of Senior Design, I was able complete testing of the application and prepare it for the public launch that will occur in the following months. I completed the final documentation and presented it to the faculty for final approval. Also I gave a demonstration of my application to faculty and the public at Tech Expo 2005.

4.2. Budget

The budget for the project is shown in the table below. All costs were existing business expenses already paid for by Valley Solutions, Inc. No new expenses were obtained during the development of this project.

Hardware			
Item	**Cost**		**Source**
Power Edge 1650	$	8,000.00	DELL (5)
Power Edge 1650	$	8,000.00	DELL (5)
Dimension 8300	$	1,100.00	DELL
Switch	$	799.00	DELL (5)
Hardware Total	**$**	**17,899.00**	
Software			
Item	**Cost**		**Source**
Visual Studio .NET 2003 Enterprise Edition	$	1,799.00	Microsoft Corporation (5)
Microsoft Action Pack Subscription*	$	300.00	Microsoft Corporation (5)
SQL Server 7.0 Standard Edition	$	1,489.00	Microsoft Corporation (5)
Software Total	**$**	**3,588.00**	
Services			
Item	**Cost**		**Source**
DNS and Network Services	$	320.00	Cogent Communications (5)
Services Total	**$**	**320.00**	
Total Budget	**$**	**21,807.00**	

*The Microsoft Action Pack Subscription provides a diverse array of applications for small businesses such as Valley Solutions, Inc. The software used for this project that is included with the Action Pack Is listed below:

Item	**Cost**	**Source**
Windows Server 2003 Enterprise Edition	--	Microsoft Corporation
Windows XP Professional w/ Service Pack 1	--	Microsoft Corporation
Office 2003 Professional Edition	--	Microsoft Corporation
SQL Server 2000 Developer Edition	--	Microsoft Corporation

5. Proof of Design

5.1. Screen Design

The Directory Web Site Management System does not have any formatting of objects on any of the pages. Any formatting of the pages are due to CSS style sheets located on the Web Site. In the following sections, the style of the pages will shown using screenshots from http://beta.truckingindex.com.

5.1.1. Client Main Menu

This is the first screen seen by clients, from here they have the option to add a new company, or view information regarding an existing company already entered in the database. They can also view their pending or rejected listings.

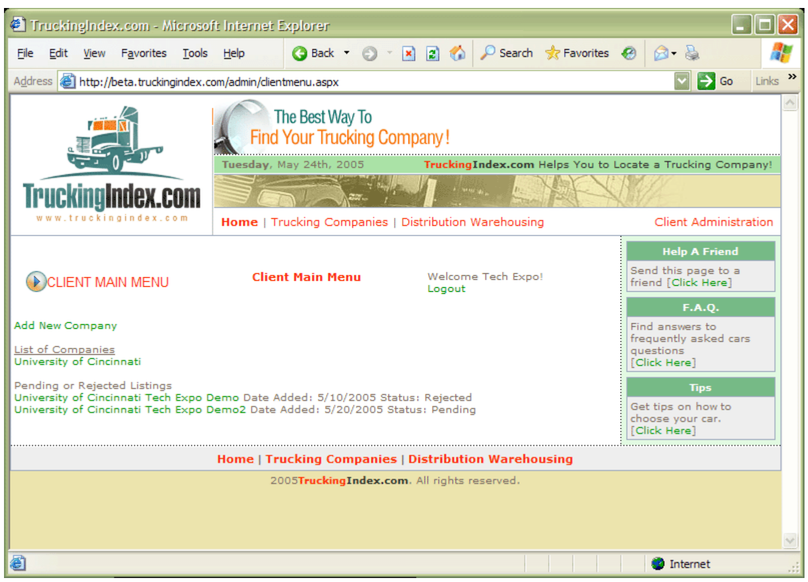


Figure 1. Client Main Screen

5.1.1.1. Company Information

On this screen, clients can view the current company information in the database. Clients have the ability to modify this information. They also have the ability to create a listing based of this information. Also on this screen, clients can view approved, pending and rejected listings.

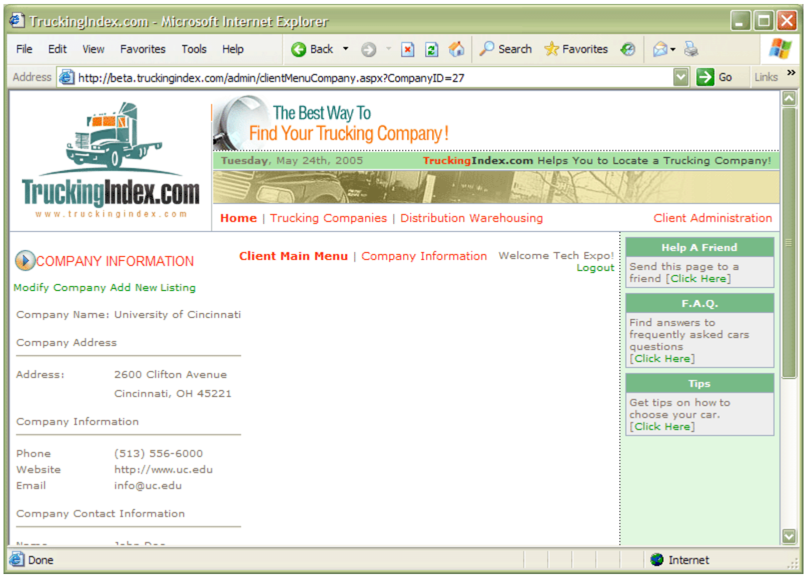


Figure 2. Company Information Screen

5.1.1.1.1. Modify Company

Using this page, clients have the ability to change their company's information.

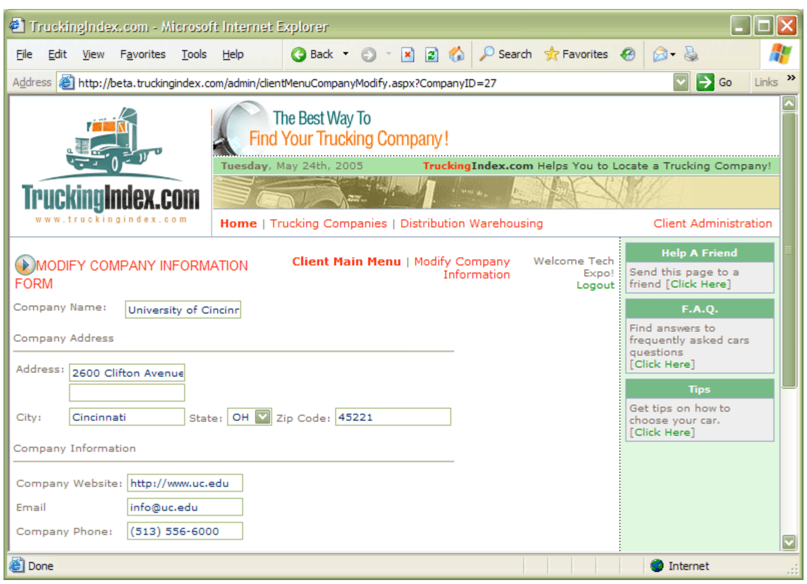


Figure 3. Modify Company Information Screen

5.1.1.1.2. Add New Listing

This screen allows clients to add a listing to the database. The listing will be placed in a pending status for administrator approval. The default information that is loaded for this screen is from the company profile. The listing information can be modified, as well as the short and long description being added at this time.

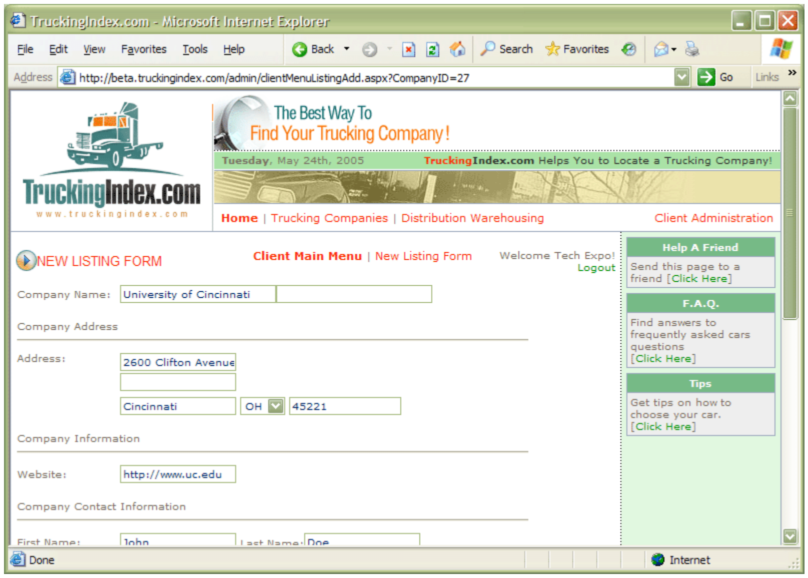


Figure 4. New Listing Form

5.1.1.1.3. Approved Listing View

This screen displays the information listed on a listing. This screen also tells what categories the listing is active on the site. Clients can modify the information for the listing, or add the listing to a category on the site.

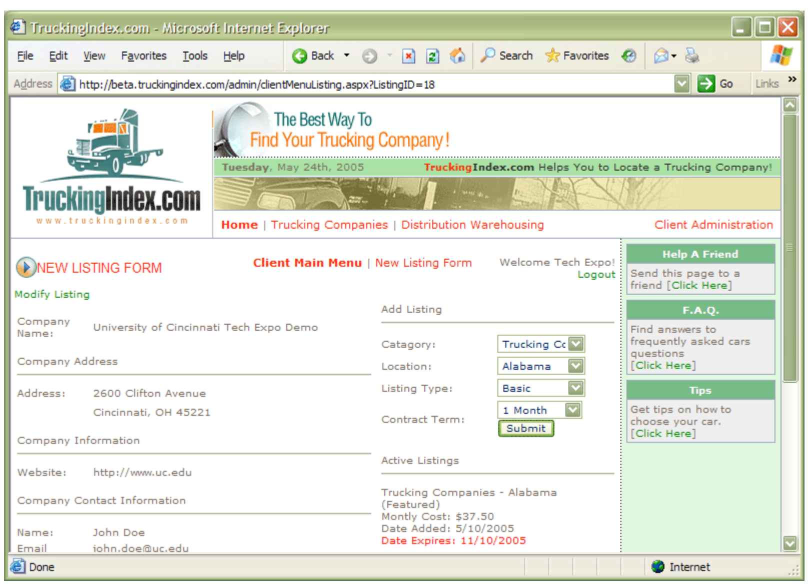


Figure 5. View Approved Listing Screen

5.1.1.1.3.1. Modify Listing

This screen allows the client to modify information in a listing, if changes are made the listing will be sent to the administrator for approval.

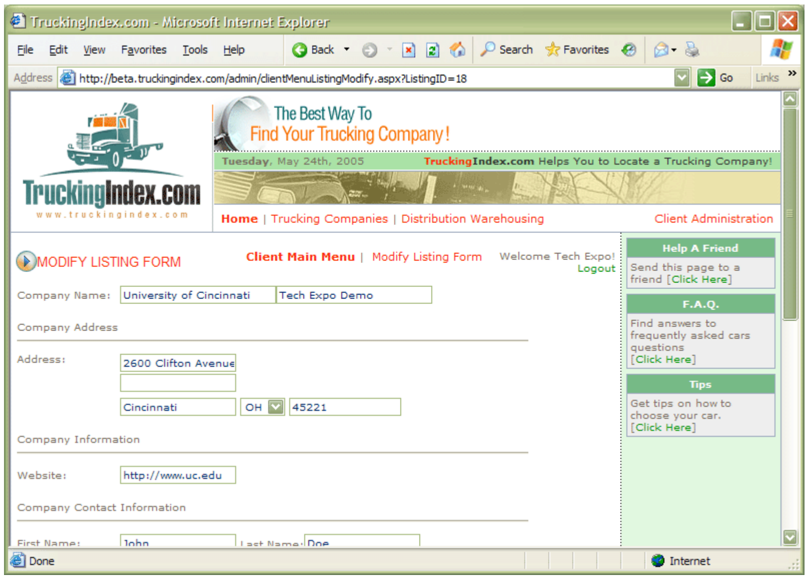


Figure 6. Modify Listing Form

5.1.1.2. Add New Company Screen

Using this form, a client will have the ability to add new companies into their profile. This is useful for end-users who may have more than one company that they run, or have divisions that may be named differently or different office locations that they might have in other cities.

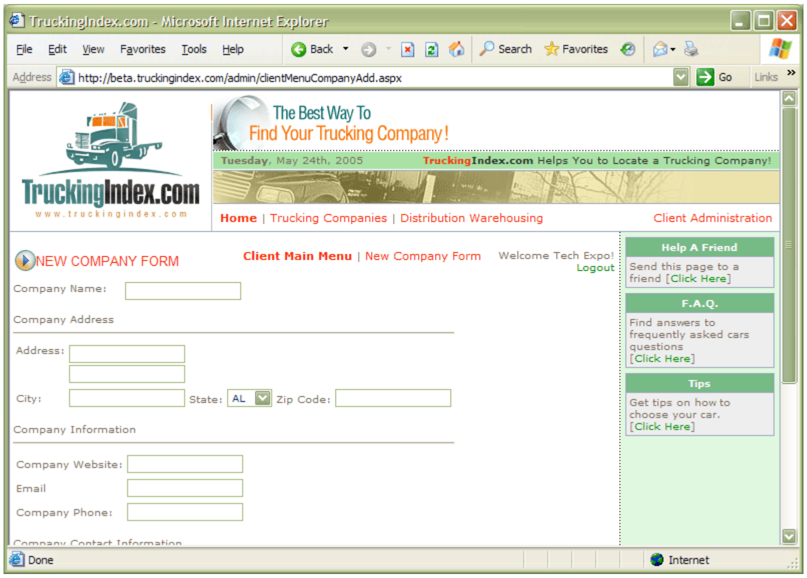


Figure 7. New Company Form

5.1.2. Administration Main Menu

This is the first screen an administrator end-user sees. The administration menu is in 3 sections: Client Administration, Site Administration, and Reporting Utilities.

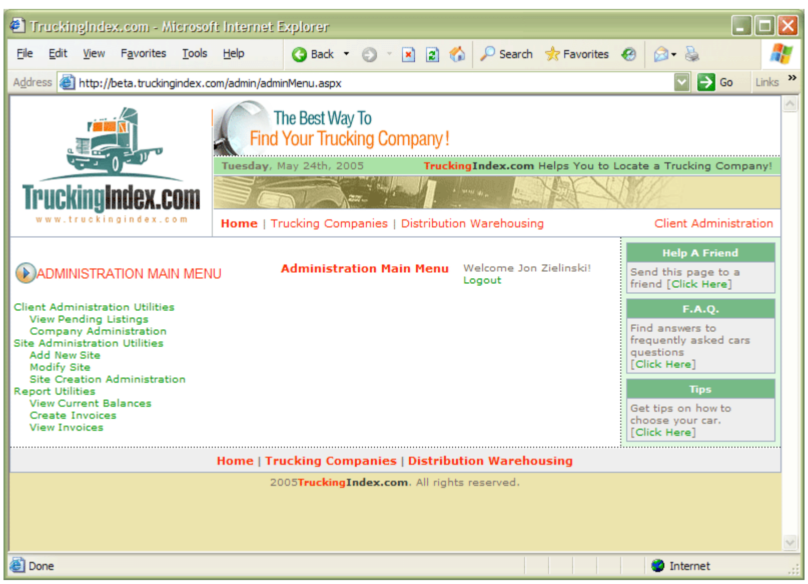


Figure 8. Administrator Main Screen

5.1.2.1. Client Administration Utilities

From this screen, administrators have the ability to view pending listings, administer company, listing and profile information.

5.1.2.1.1. View Pending Listings

This screen will display pending listings for all sites using the Directory Web Site Management System. Administrators will have the ability to either approve or reject the listing.

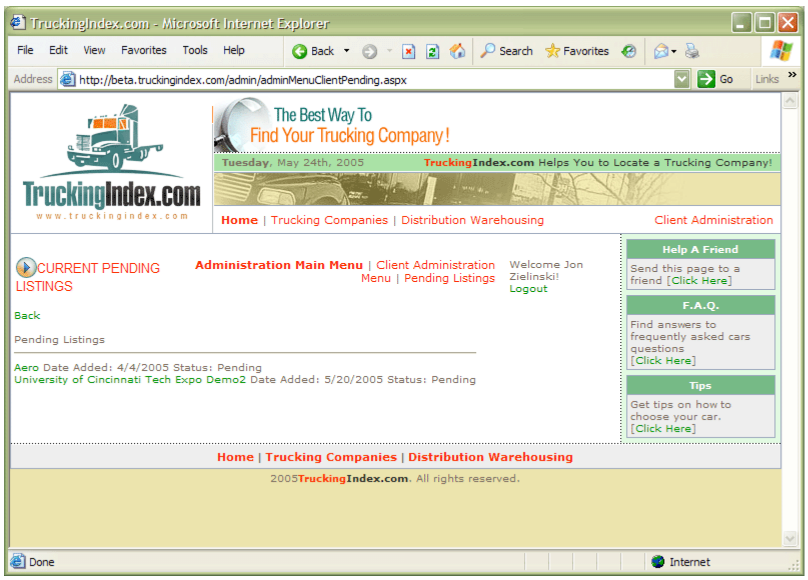


Figure 9. Pending Listing Screen

5.1.2.1.2. Company Administration

This screen shows the active companies in the database, administrators can click the company name to view/modify the company information. Or go to another page to view active listings listed for this company.

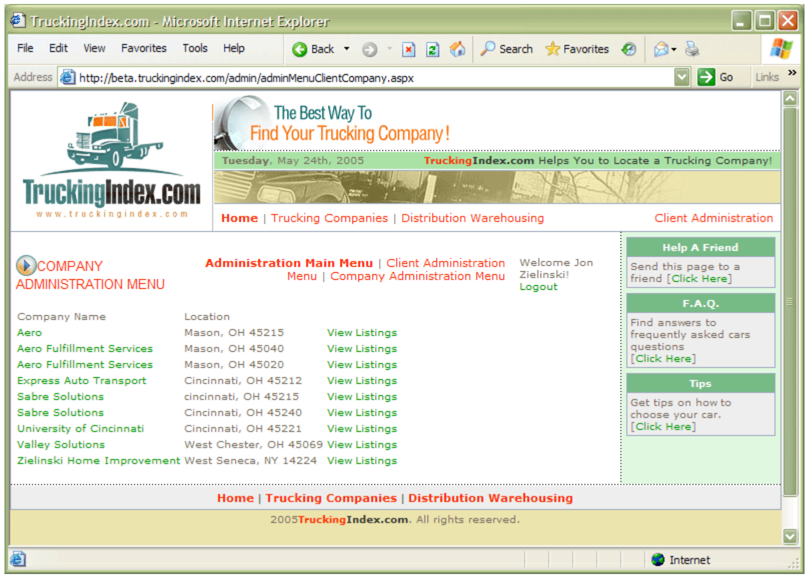


Figure 10. Company Administration Main Screen

5.1.2.1.2.1. View/Modify Company Information

This screen allows the administrator to modify information on a particular company if changes are needed to be made.

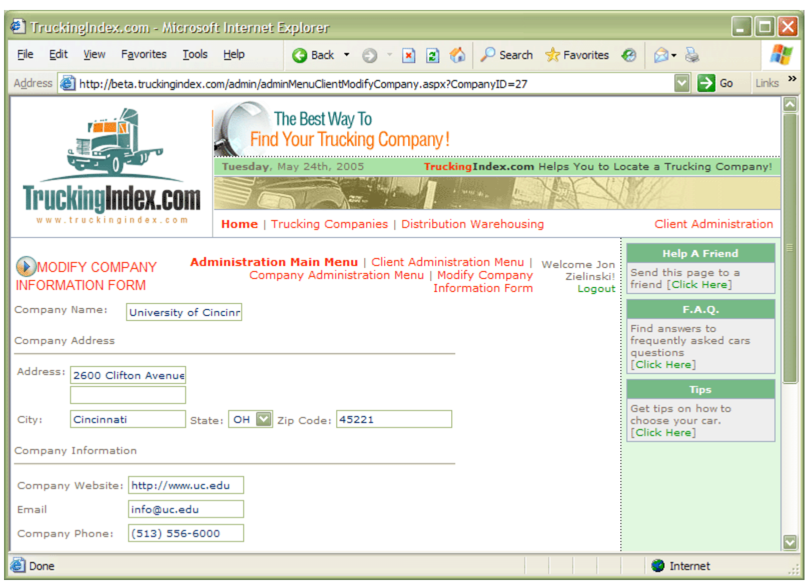


Figure 11. Modify Company Information Screen

5.1.2.1.2.2. View Approved Listings

This screen displays the information listed on a listing. This screen also tells what categories the listing is active on the site. Administrators can modify the information for the listing, or add the listing to a category on the site.

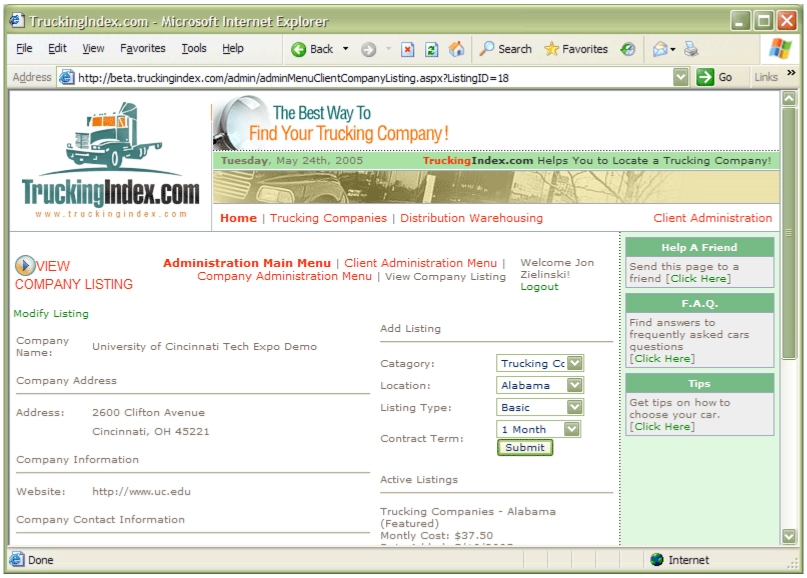


Figure 12. Approved Listing Main Screen

This screen also allows the administrator to view listings for a particular company. They can click the listing to view detailed information.

5.1.2.1.2.2.1 Modify Listing

This screen allows the administrator to modify information in a listing, if changes are made the listing will be sent to the administrator for approval.

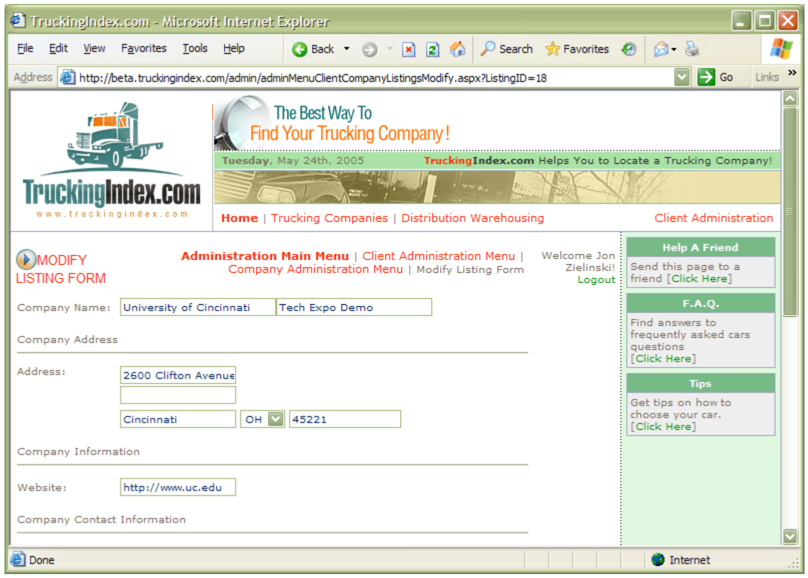


Figure 13. Modify Listing Screen

5.1.2.1.3. View Pending Listing(s)

This screen shows the pending listings. If a listing is pending, the administrator can view the listing and approve or reject the listing.

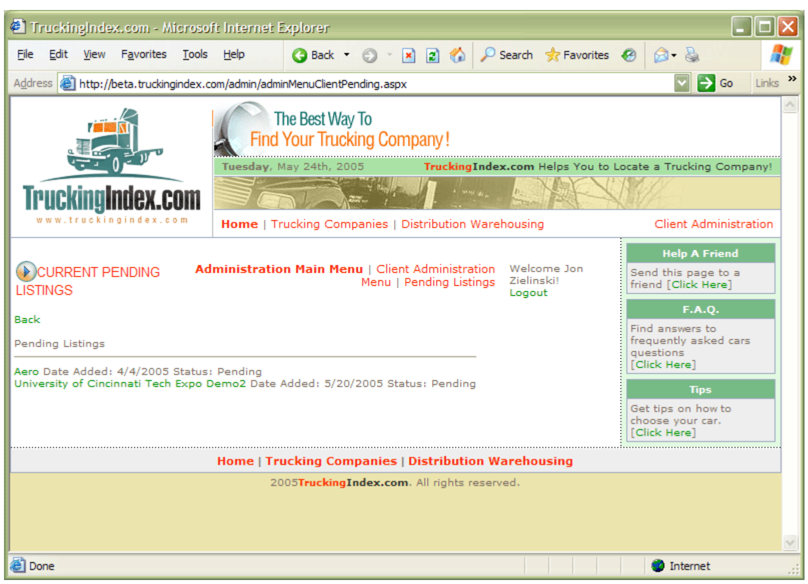


Figure 14. Pending Listing Screen

5.1.2.2. Site Administration

This screen allows the administrator to perform different site related actions. The site administration section allows the administrator to add, modify and remove Web sites using the application. They can also add, remove or modify the categories available on that site. This area would also have the location of template files used in creating the files.

5.1.2.2.1. Add New Site

This page will allow the administrator to add Web Site information to be used for a new site. This information includes the location of template files, URL, and the physical location of the Web site.

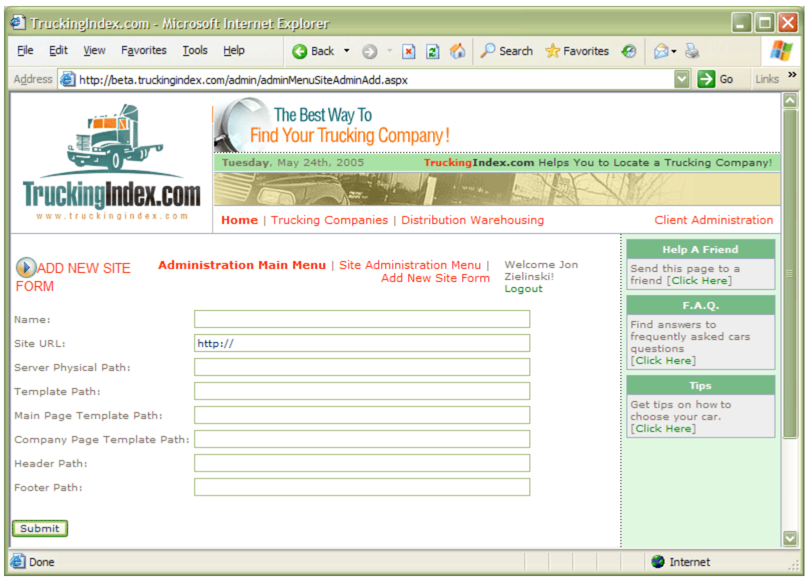


Figure 15. Add New Site Form

5.1.2.2.2. Modify Site Information

This page will allow the administrator to modify information related to existing Web sites set up to use the project. The information that can be changed includes the location of template files, URL, and the physical location of the Web site.

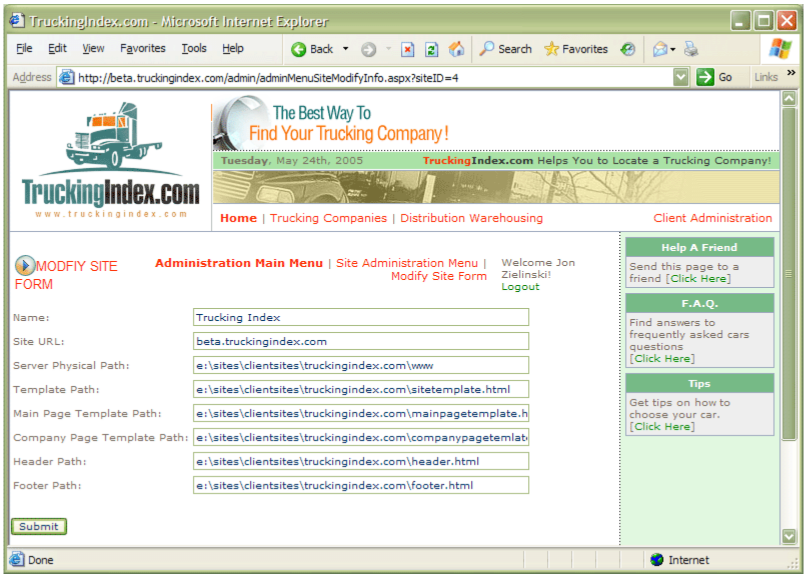


Figure 16. Modify Site Information Form

5.1.2.2.2.1. Manage Categories Page

Using this page, administrators will be able to modify category information on a particular site by clicking a link to take you to the proper form. Also, using this page will allow the administrator to add categories to a site. Information used includes a category name, if the category has a parent category or not, and the cost of listing on this category.

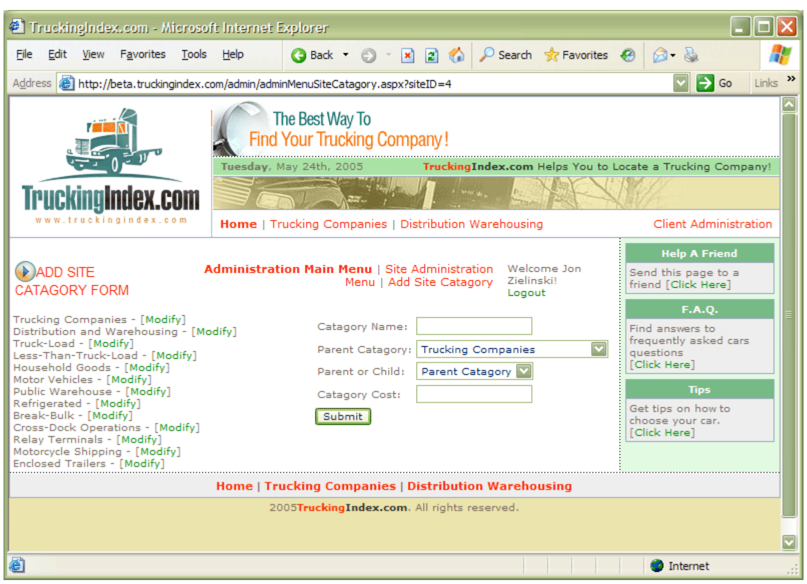


Figure 17. Manage Site Categories Screen

5.1.2.2.2.2. Modify Category Page

Using this page will allow the administrator to modify category information. Information that can be modified includes a category name, if the category has a parent category or not, and the cost of listing on this category.

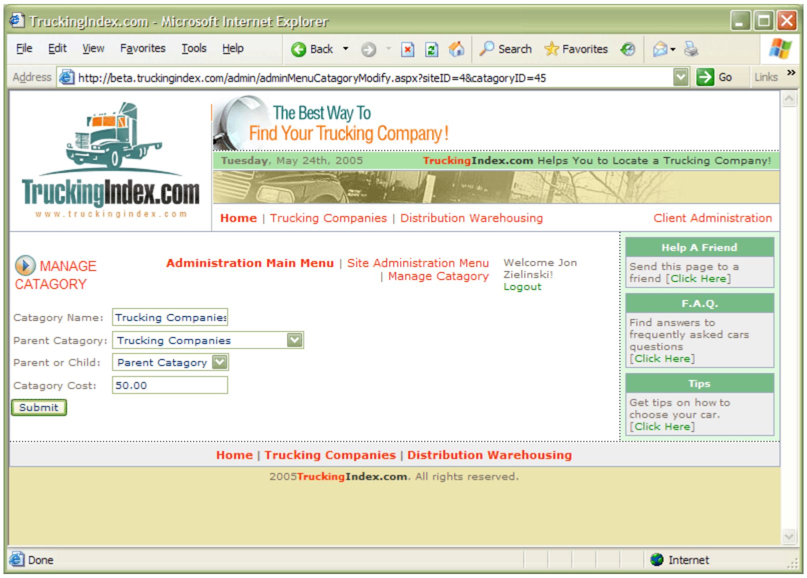


Figure 18. Modify Category Form

5.1.2.2.3. Site Creation Utility

The administrator can use this screen to manually create the directory pages, and company pages.

5.1.2.3. Report Utilities

5.1.2.3.1. Create Invoices

This link will call a stored procedure and create the invoice information. When completed, the administrator will return to the Report Utilities menu.

5.1.2.3.2. View Invoice

Once an invoice is created, the basic information will be displayed here. The information displayed will include: Company's Name, Location, amount due, status of the invoice, and the date the invoice was created. You can also view more detailed information by clicking the “view details” link located near each invoice. Finally you will have the ability to E-mail any listings individually at are at a ready status.

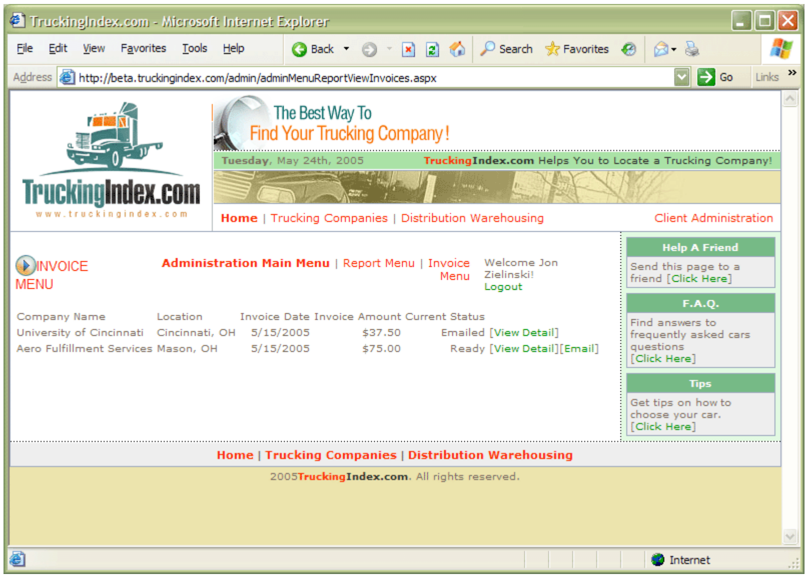


Figure 19. Invoice Main Screen

The status of the invoice could be one of the following: ready and E-mailed. When the invoice is first created it is in a status of ready, meaning it was ready to be E-mailed. A status of E-mailed indicates that the invoice has been sent to the appropriate client via E-mail.

5.1.2.3.2.1. View Invoice Detail

This page will display detailed invoice information. The information displayed on this page will be per listing and will include: Web site, listing company name, location, category page, and cost of the listing.

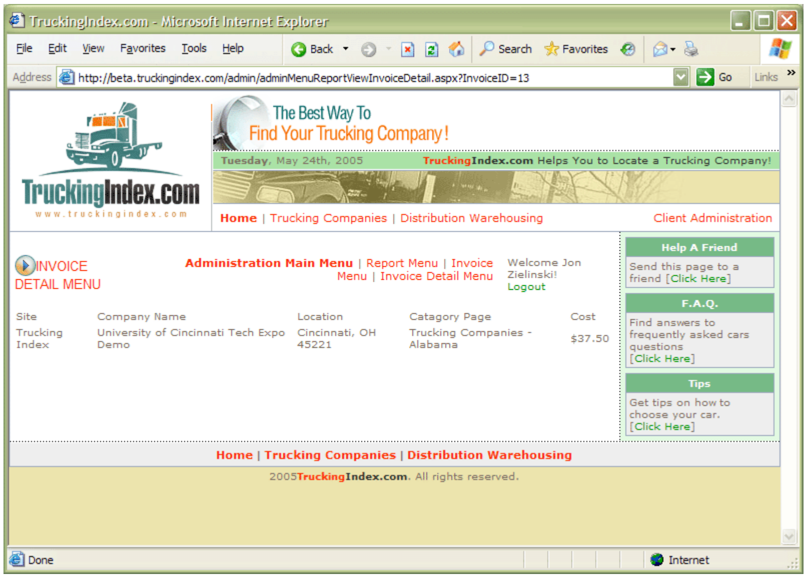


Figure 20. Invoice Detail Screen

5.1.2.3.2.2. E-mail Invoice

Using this link will E-mail all invoices that are in a ready status for the current month.

5.1.2.3.2.3. View Current Balances

This page will display the current balance of each account using the project. Information displayed here includes the Company name, location, and the current balance of that account. A link is provided to view detailed transaction information. When an invoice is created, a transaction record will automatically be added with the appropriate information.

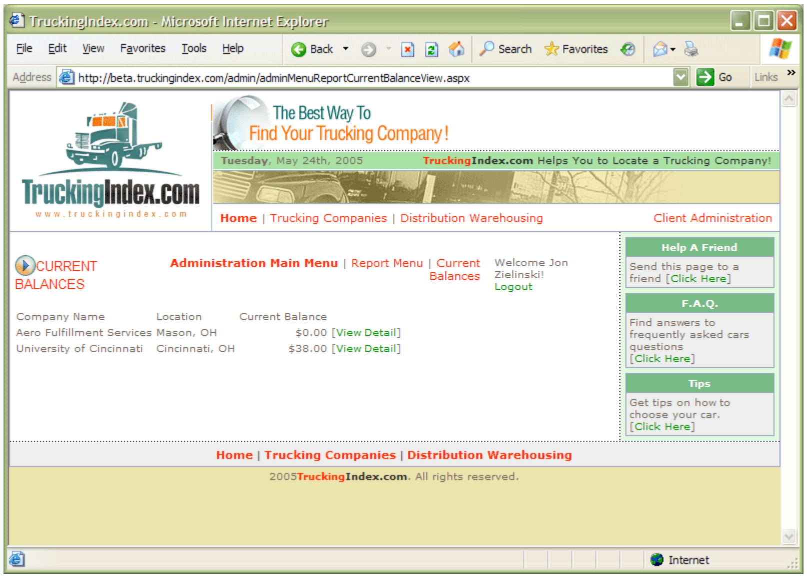


Figure 21. View Current Balances Screen

5.1.2.3.2.4. Transaction Detail

This page will display financial transactions that the selected account occurs. Types of transactions include bills, payments and credits. A link is provided to add a transaction record. Here a form is used to monitor payments or credits issued to a account.

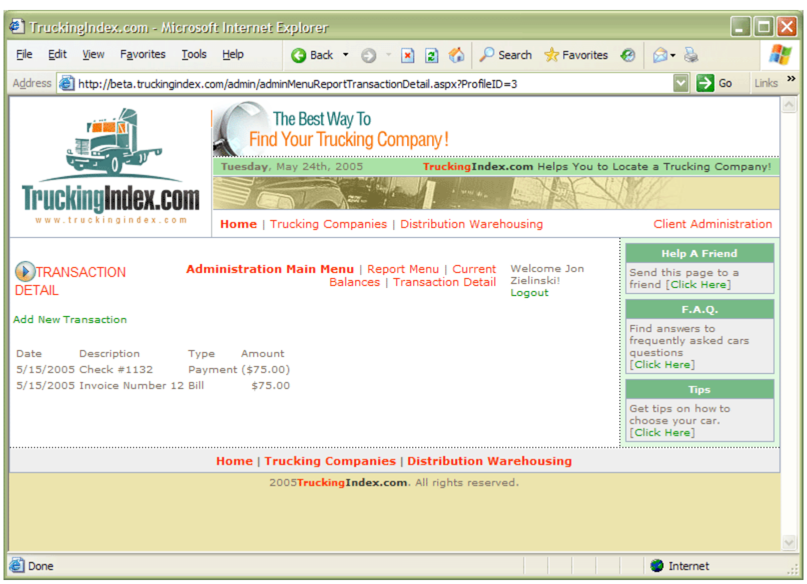


Figure 22. Transaction Screen

5.1.3. Created Pages

The project will create two types of pages, the directory and company page. If needed, the project will also create the directory structure needed for the particular page.

5.1.3.1 Directory Pages

Directory pages will be populated with featured and/or basic listings. Examples can be seen below. These pages will be created using an existing template. Various portions of the template page will be replaced and populated when the HTML file is created.

Valley Solutions, Inc. (513) 755-7766
Featured In **Webpage | Company Info | E-mail**
Valley Solutions, Inc. is a privately held corporation based in West Chester, Ohio. We provide a diverse range of products and services that are based on our three core competencies of: Web Design, Search Engine Optimization and Custom Programming.

Figure 23. Sample Featured Listing

Valley Solutions, Inc. (513) 755-7766

Figure 24. Sample Basic Listing

5.1.3.2 Company Pages

Like the category pages, company pages will be created using a template file. Information used on the HTML page will be inserted when the page is created.

Company Information

Valley Solutions, Inc.
Valley Solutions, Inc. is a privately held corporation based in West Chester, Ohio. We provide a diverse range of products and services that are based on our three core competencies of: Web Design, Search Engine Optimization and Custom Programming.

Webpage:	http://www.valleysolutionsinc.com
Contact Name:	Andrew Wash
E-mail:	Andrew@valleysolutions.com
Phone:	(513) 755-7766

Valley Solutions, Inc.

8903 Cox Road
West Chester, OH 84069

Valley Solutions, Inc. is a privately held corporation based in West Chester, Ohio. We provide a diverse range of products and services that are based on our three core competencies of: Web Design, Search Engine Optimization and Custom Programming.

Figure 25. Sample Company Information

5.2. Database Design

The following pages will display the database design for the project. Due to the size of the database, I have broken the schema into three smaller sections.

5.2.1. Company and Listing Schema

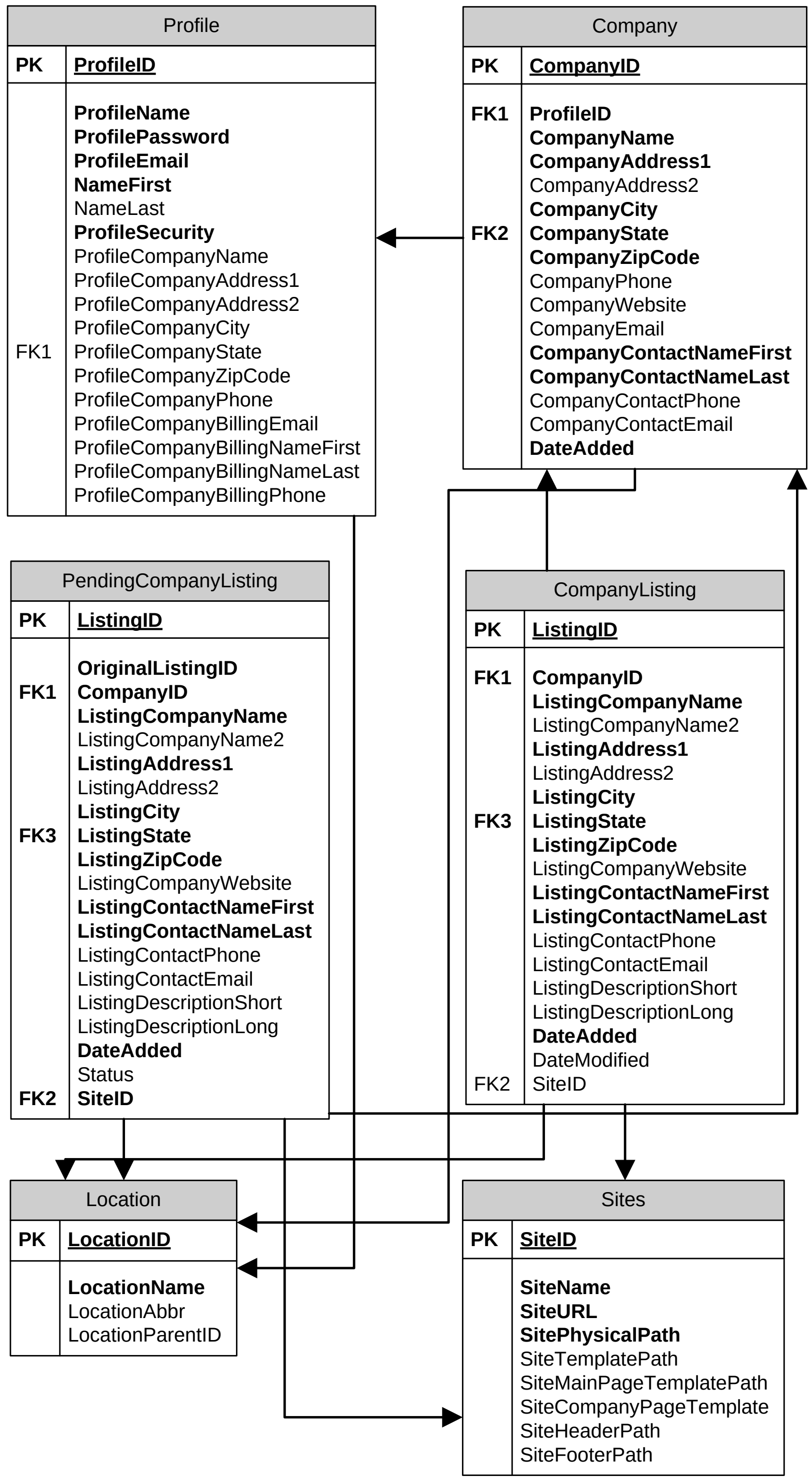


Figure 26. Database Schema

5.2.2. Category Listing Schema

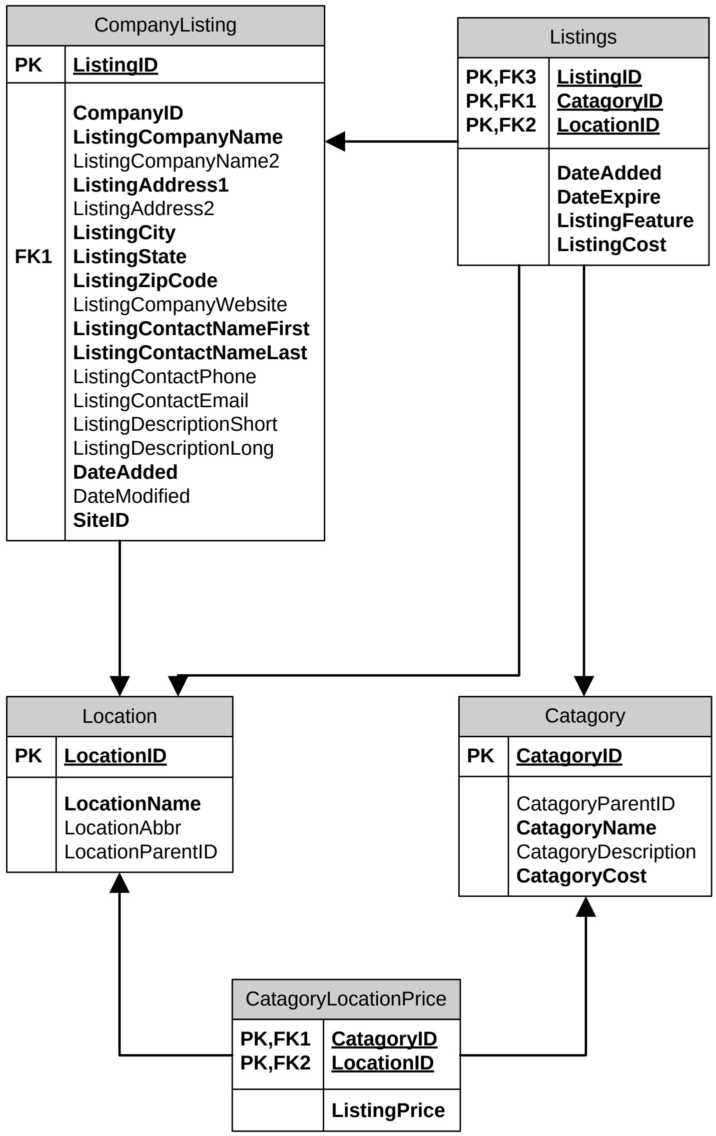


Figure 27. Database Schema

5.2.3. Invoice and Transaction Schema

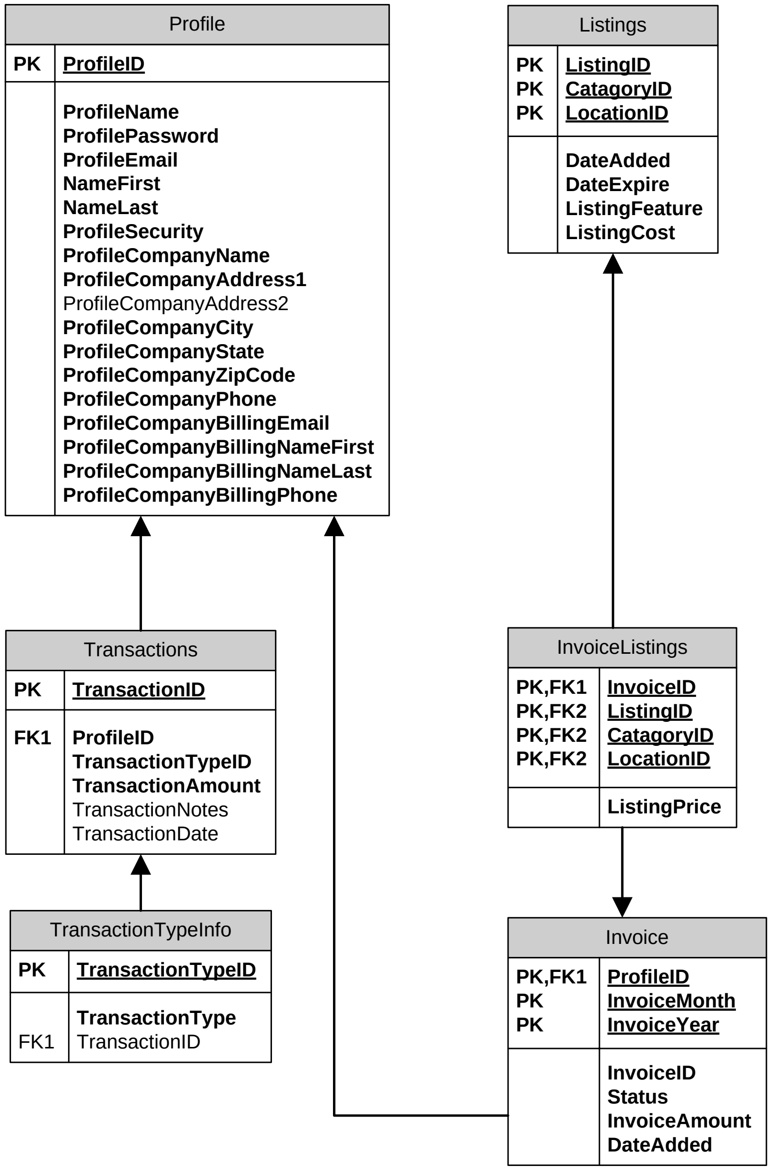


Figure 28. Database Schema

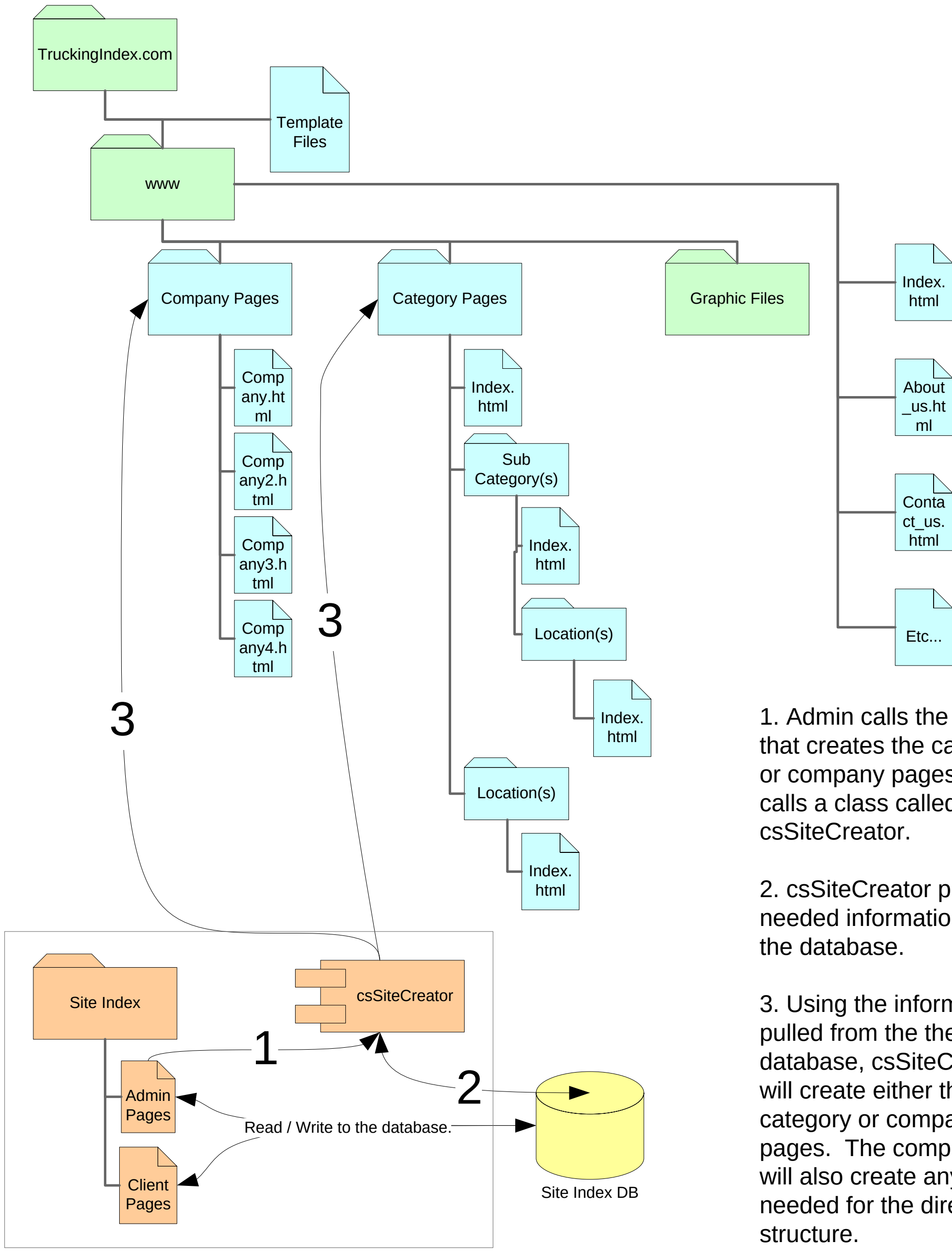
5.3. Physical and Virtual Directory Structure

The following sections show how the physical and virtual directory structures are set up. Not all directories and files are represented. For instance, the number of category directories could vary from site to site. This could also be said for the company pages.

5.3.1. Physical Directory Structure

The following chart is a high level representation of how the physical directory structure is laid out on the project. There could be many more directories listed for individual sites, but for simplicity, only TruckingIndex.com is currently shown. Also in this graph is a high level explanation of how the administrator calls on class to create the directories and files needed in the directory structure.

SiteIndex is in its own physical directory. Separate from TruckingIndex.com or any other site that would use the Directory Web Site Management System. Again for simplicity, I have shown the pages as admin and client pages. The pages contained in those groupings are explained in the sections above. The administrator has the ability to call a class that will create the folders and files for the company and category pages of a particular site, in this case TruckingIndex.com. The Directory Web Site Management System only creates the directories and/or files for the company and/or category directories. Other files or directories under TruckingIndex.com would be managed by the Webmaster of the site or server. Also, for simplicity, I have represented that the template files are located under TruckingIndex.com, when in reality, I have set up the application so that the template files could located anywhere on a physical or networked drive.



1. Admin calls the page that creates the category or company pages. This calls a class called csSiteCreator.

2. csSiteCreator pulls the needed information from the database.

3. Using the information pulled from the the database, csSiteCreator will create either the category or company pages. The component will also create any folders needed for the directory structure.

Figure 29. Physical Directory Diagram

5.3.2. Virtual Directory Structure

Below is a graph showing how a Web Site calls on the Site Index directory as a virtual directory. Any site set up to use the Directory Web Site Management System would call on the same physical directory using a virtual directory in IIS.

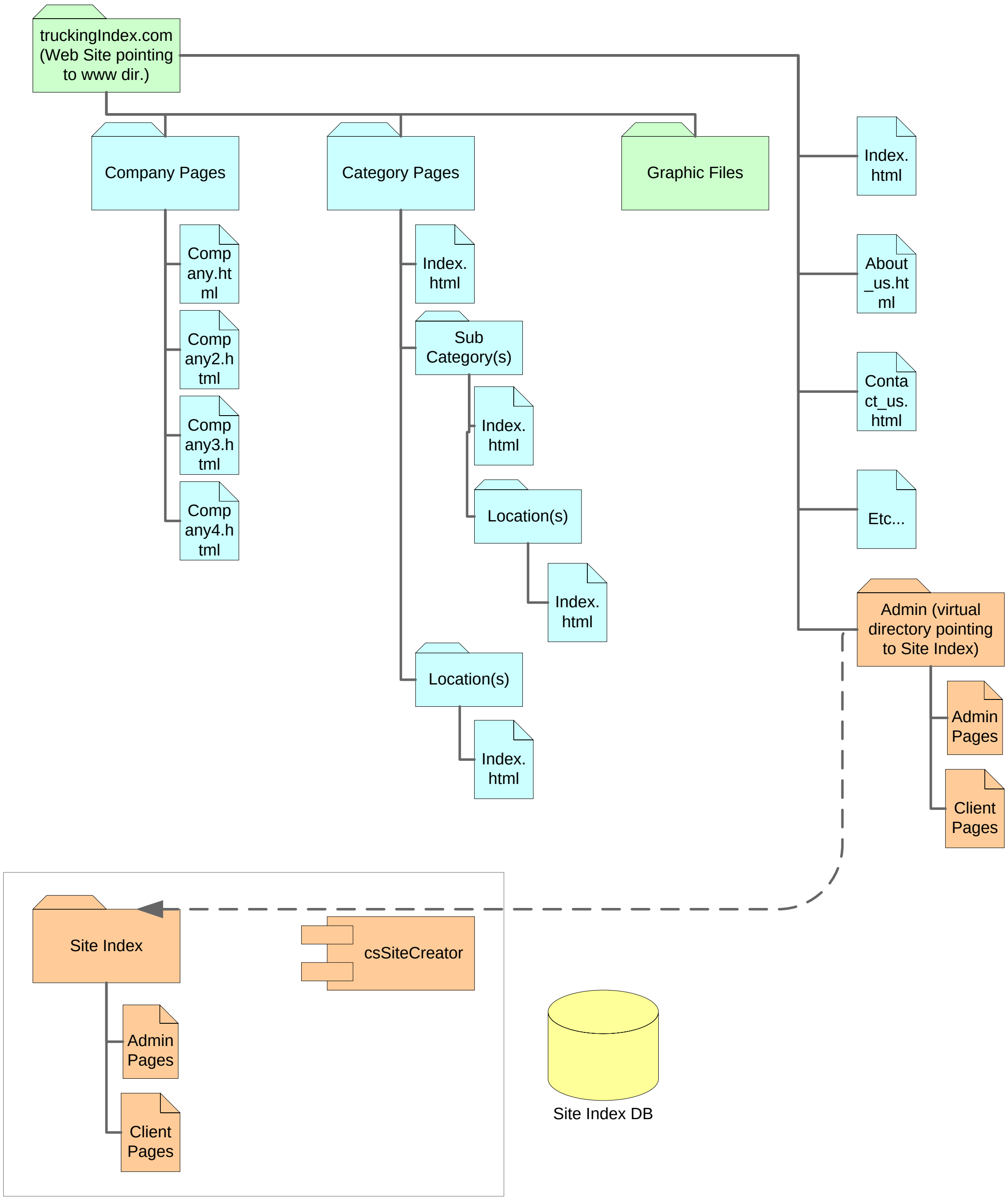


Figure 30. Virtual Directory Diagram

6. Testing Procedures

6.1. Developer Testing

After completing each page or component, test data was submitted by the developer. If the desired output was not obtained, modifications to the code would be made until the desired effect could be reached.

6.2. Prototype Testing

Once a prototype was completed, further testing was conducted by the developer and client. Data was entered simulating information that a potential client of Valley Solutions would enter. Bugs that were discovered would be tracked and resolved by the developer.

6.3. Beta Testing

Beta testing using existing clients of Valley Solutions, Inc. will begin towards the end of Senior Design III. Current plans are to move the Directory Web Site Management System from beta to live beginning in June. Valley Solutions, Inc. is choosing TruckingIndex.com as the first site to go live with the Directory Web Site Management System. They will invite existing clients who currently list on other Valley Solutions sites to participate in the beta testing. Valley Solutions will set up an E-mail address for clients to report any bug not found during the prototype beta testing.

7. Conclusions and Recommendations

7.1. Conclusions

This project was created in response to Valley Solutions, Inc need for an updated backend to manage multiple Web sites they own. To prepare the project I used Microsoft

Visual Studio.NET 2003, Microsoft SQL Server 2000, and ASP.NET using C#. The project was completed over the three quarter Senior Design sequence. The budget of approximately $22,000 would be a real-world estimate for the completion of this project before labor costs. The project fulfilled all Design Freeze deliverables. Testing was performed to ensure the product's usability.

7.2. Recommendations

During the beginning of this project, I had never coded in ASP.NET before. I was not aware of all the differences between ASP 3.0 and ASP.NET. My first issue came after I found out how to bind Data Reader from a database to a drop down control on a form. At first I was coding the way I knew how, manually populating the control. After binding the control, there was information in the database that I used to test previously that caused me to break references. Thankfully for me, I was in a prototype stage of development and the fix was easy enough to do. Had this project gone live, and clients had information in the database that would have eventually caused an error, the situation could have been worse. Had I not tested the fix to make sure that it worked or did not work with specific data; this issue would have caused major headache later.

Based on this, I recommend that when ever you change even a small thing such as binding a drop down list, make sure you have the ability to test every possible peace of data that the drop down list may or may not encounter. In my case, I had nulls in my database and when I tried selecting a null value on the drop down, my code broke. This can occur when trying to bind any type of control.

Another area that caused me grief is when I attempted to figure out the best way to reference the template files in the project. Currently I am saving a physical path (c:\example_directory\template.html) in the database as a varchar(255). I look to see if the file exists, and if it does, I read it in to a string variable in the project. From here I can replace data that I choose to replace. The other method that I attempted to use that caused problems was storing the template data in the database as a text value. This proved to be more difficult than it really needed to be. I could view the data I put in the database, and displaying the data in a textbox control also provided some challenges.

It is my recommendation that when you use large files, that you reference files outside of your database. If you need to modify them, it is easier to do so in a browser that is designed to do so already. In my cased, I can open the HTML template files in Visual Studio. If I kept the files in the database, and eventually figured out how to do everything I truly wanted to do with the data, I could have reinvented the wheel in a sense.

Appendix A.

Bug Testing Results

Bug #	Date Reported	Issue	Date Resolved	Resolution
1	4/3/05	When modifying a pending listing on clientMenuListingModify.aspx, only a single character is saved to the Long Description field.	5/15/05	This was due to me not specifying the number of characters in the stored procedure. Changed the value to what is defined in the table and the update now works fine.
2	4/3/05	When adding an approved listing on clientMenuListing.aspx, the contract length is adding days and not months.	5/15/05	Fixed the insert query by adding datepart month to the date.
3	4/3/05	When modifying an approved listing, if there is already a pending listing for that approved listing; the program will insert a new pending listing instead of updating the current pending listing for the same listing. Need to check with Drew to determine if this is fine, or if he wants to update the existing pending listing.	5/15/05	Resolved by changing the query that updates the modified information.
4	4/3/05	Company information does not display on adminMenuClientModifyCompany.aspx.	4/9/05	Had to recreate the page for some reason. The code behind was not getting called at all even though the correct file name was used. Now working as it should.
5	4/3/05	When modifying a rejected listing, the status of the listing does not change to pending.	4/25/05	Modified update query to also update the status of the pending listing.
6	4/3/05	The Long Description field on clientMenuListingModify.aspx is not populated by the information that is in the database.	4/9/05	Issue was caused by not populating the text control. Added the proper code and it was resolved easily.

Bug #	Date Reported	Issue	Date Resolved	Resolution
7	4/3/05	Pending and/or Rejected Listings do not show up on adminMenuClientCompanyListings.aspx.	4/9/05	The proper query was not being called, thus not showing the correct data.
8	4/8/05	Category drop down is not populating after Session["SiteID"] change on clientMenuListing.aspx.	4/9/05	The wrong session variable was being used in place of the SiteID variable. Corrected the error and the drop down is populating correctly now.
9	3/29/05	The code is breaking when trying to select a value on the location list on an existing listing.	4/2/05	This was the result of a change from the prototype to the beta version of the project. In the prototype version, I was populating dropdown lists manually which included a 0 value representing no location.. I made a change to bind the dropdown list to a record set that pulled from the location table. Since there is no record id 0, this value was not available on the dropdown list. I resolved the issue by updating all the records from the prototype testing that had a location of 0 to a number that was available. There were no more errors from that point.
10	5/16/05	After I added form validation, when the form had a missing value and did not complete the task that it should finish, the login area would add to itself repeating the same phrase. I.E. it would say "Welcome Jon Zielinski!Welcome Jon Zielinski!Welcome Jon Zielinski!".	5/16/05	This bug was caused by adding text to already existing text in a textbox. This was done by using "+=". Changed it to "=" on all of the pages and this resolved the issue.

Appendix B.

Code Snippets

B 1. Header and Footer Literal Placeholders Code

One of the key features of the project not listed in the deliverables is the ability to be used with multiple Web sites. In ASP 3.0 this was done with server side includes. ASP.NET does this differently. Well there are more than one way to do this in ASP.NET actually. The way I accomplished this was with the use of literal placeholders. Placeholders are like labels, but they display straight text rather than formatted text.

In ASP 3.0, the code would have looked like this.

```
<!-- #include file="./header.asp" -->
<!-- #include file="./footer.asp" -->
```

In ASP.NET, this same task was accomplished in a few steps. The first step is to put the literal place holder tags on the aspx file.

```
<asp:literal id="header" runat="server"></asp:literal>
<asp:literal id="footer" runat="server"></asp:literal>
```

Now that the tags are on the aspx file, on the code behind you can assign text to literal placeholders. In this case, I passed the path to the header and footer files to a function that will return the text that I am looking for. I did it this way so that I did not need to store a large amount of text in the database, rather the path to the files I wanted. So I am passing a path as a string like “c:\truckingindex.com\header.html”.

```
this.header.Text =
mySiteCreator.getPageTemplate(Session["SiteHeaderPath"].ToString());
this.footer.Text =
mySiteCreator.getPageTemplate(Session["SiteFooterPath"].ToString());
```

This is the last step to return the header or footer file. The function receives the string of the path to the desired file. Then it opens the file and reads it line by line and assigns it to a string. Once the file is read, then it returns the string and assigns it to the literal placeholder.

```
public string getPageTemplate(string TemplatePath)
{
	string PageTemplate = "";
	StreamReader reader = File.OpenText(TemplatePath);

	try
	{
		string str;
		str = reader.ReadLine();
		while (str != null)
		{
			PageTemplate += str;
			str = reader.ReadLine();

		}	// end while
	}	// end try

	catch (Exception error)
	{
		string strError = "";

		strError += error.Message;
		if (error.InnerException != null)
			strError += error.InnerException.Message;
		System.Diagnostics.EventLog.WriteEntry("SiteIndex",
strError, System.Diagnostics.EventLogEntryType.Warning);

	}	// end catch

	finally
	{
		reader.Close();

	}	// end finally

	return PageTemplate;

}	// end getPageTemplate
```

B 2. Creating the Company Information Pages

To create the company information pages, when the administrator clicks the link to create the pages for a specific Web site. This calls on the createCompanyPages subroutine. This will call the database and retrieve some information needed for the pages, most specifically where the company pages need to be created on the network. A location similar to “c:\truckingindex.com” is retrieved from the database, then passed to another subroutine with the SiteID

```
public void createCompanyPages(int SiteID)
{
	SqlConnection sqlConn = new 
SqlConnection(mySiteIndex.strSQLConnection);
	SqlDataReader sqlRdr = null;

	try
	{
		sqlConn.Open();
		SqlCommand sqlComm = sqlConn.CreateCommand();
		sqlComm.CommandText = "sp_getSiteInfo";
		sqlComm.CommandType = CommandType.StoredProcedure;
		sqlComm.Parameters.Add("@SiteID", SqlDbType.Int);
		sqlComm.Parameters[0].Value = SiteID;
		sqlRdr = sqlComm.ExecuteReader();

		if (sqlRdr.HasRows)
		{
			while (sqlRdr.Read())
			{
				SitePhysicalPath = 
sqlRdr["SitePhysicalPath"].ToString();
				SiteCompanyPageTemplate = 
getPageTemplate(sqlRdr["SiteCompanyPageTemplate"].ToString());
				populateCompanyPages(sqlRdr["SitePhysicalPath"]
			.ToString(), SiteID);

			}	// end while
		}	// end if
	}	// end try

	catch (Exception error)
	{
		string strError = "";

		strError += error.Message;
		if (error.InnerException != null)
			strError += error.InnerException.Message;
			System.Diagnostics.EventLog.WriteEntry("SiteIndex", 
strError, System.Diagnostics.EventLogEntryType.Warning);
	}	// end catch

	finally
	{
		sqlRdr.Close();
		sqlConn.Close();
	}	// end finally
```

```
}	// end createCompanyPages
```

PopulateCompanyPages takes the Path string and the site ID and queries the database. It also reads the Company Information Template file and assigns it to a string. It then goes on to replace certain comments in from template file string with information from the database, then writes that string to a file.

```
public void populateCompanyPages(string Path, int SiteID)
{
	SqlConnection sqlConn = new
SqlConnection(mySiteIndex.strSQLConnection);
	SqlDataReader sqlRdr = null;

	try
	{
		sqlConn.Open();
		SqlCommand sqlComm = sqlConn.CreateCommand();
		sqlComm.CommandText = "sp_adminMenuSiteCreateCompany";
		sqlComm.CommandType = CommandType.StoredProcedure;
		sqlComm.Parameters.Add("@SiteID", SqlDbType.Int);
		sqlComm.Parameters[0].Value = SiteID;

		sqlRdr = sqlComm.ExecuteReader();

		string companyFolderPath = Path + "\\company_info\\";
		Directory.CreateDirectory(companyFolderPath);
		if (sqlRdr.HasRows)
		{
			while (sqlRdr.Read())
			{
				string companyPagePath = Path +
"\\company_info\\" + sqlRdr["CompID"].ToString() + "_" +
sqlRdr["ListID"].ToString() + ".html";
				string CompanyPage = SiteCompanyPageTemplate;

				//  <!-- COMPANYNAME-->
				CompanyPage = CompanyPage.Replace("<!--
COMPANYNAME -->", sqlRdr["ListingCompanyName"].ToString());
				//  <!-- COMPANYNAME2-->
				CompanyPage = CompanyPage.Replace("<!--
COMPANYNAME2 -->", sqlRdr["ListingCompanyName2"].ToString());
				//  <!-- SHORTDESCRIPTION -->
				CompanyPage = CompanyPage.Replace("<!--
SHORTDESCRIPTION -->", sqlRdr["ListingDescriptionShort"].ToString());
				//  <!-- COMPANYWEBPAGE-->
				CompanyPage = CompanyPage.Replace("<!--
COMPANYWEBPAGE -->", sqlRdr["ListingCompanyWebsite"].ToString());
				//  <!-- CONTACTNAMEFIRST-->
				CompanyPage = CompanyPage.Replace("<!--
CONTACTNAMEFIRST-->", sqlRdr["ListingContactNameFirst"].ToString());
				//  <!-- CONTACTNAMELAST-->
```

```
                        CompanyPage = CompanyPage.Replace("<!--
CONTACTNAMELAST -->", sqlRdr["ListingContactNameLast"].ToString());
                        //  <!-- CONTACTEMAIL-->
                        CompanyPage = CompanyPage.Replace("<!--
CONTACTEMAIL -->", sqlRdr["ListingContactEmail"].ToString());
                        //  <!-- CONTACTPHONE-->
                        CompanyPage = CompanyPage.Replace("<!--
CONTACTPHONE -->", sqlRdr["ListingContactPhone"].ToString());
                        //  <!-- COMPANYADDRESS1-->
                        CompanyPage = CompanyPage.Replace("<!--
COMPANYADDRESS1 -->", sqlRdr["ListingAddress1"].ToString());
                        //  <!-- COMPANYADDRESS2-->
                        if (sqlRdr["ListingAddress2"].ToString() != ""
|| sqlRdr["ListingAddress2"].ToString() != null)
                              CompanyPage = CompanyPage.Replace("<!--
COMPANYADDRESS2 -->", sqlRdr["ListingAddress2"].ToString());
                        //  <!-- COMPANYCITY-->
                        CompanyPage = CompanyPage.Replace("<!--
COMPANYCITY -->", sqlRdr["ListingCity"].ToString());
                        //  <!-- COMPANYSTATE-->
                        CompanyPage = CompanyPage.Replace("<!--
COMPANYSTATE -->", sqlRdr["LocationAbbr"].ToString());
                        //  <!-- COMPANYZIPCODE-->
                        CompanyPage = CompanyPage.Replace("<!--
COMPANYZIPCODE -->", sqlRdr["ListingZipCode"].ToString());
                        //  <!-- LONGDESCRIPTION -->
                        CompanyPage = CompanyPage.Replace("<!--
LONGDESCRIPTION -->", sqlRdr["ListingDescriptionLong"].ToString());
                        //  <!-- TITLE -->
                        CompanyPage = CompanyPage.Replace("<!-- TITLE -
->", sqlRdr["ListingCompanyName"].ToString());

                        StreamWriter writer = new StreamWriter
(companyPagePath, false);
                        writer.WriteLine(CompanyPage);
                        writer.Close();

                  }     // end while
            }     // end if
      }     // end try

      catch (Exception error)
      {
            string strError = "";

            strError += error.Message;
            if (error.InnerException != null)
                  strError += error.InnerException.Message;

            System.Diagnostics.EventLog.WriteEntry("SiteIndex",
      strError, System.Diagnostics.EventLogEntryType.Warning);
      }     // end catch

      finally
      {
            sqlRdr.Close();
            sqlConn.Close();
```

```
      }     // end finally
}     // end createCompanyPages
```

B 3. Invoice Email Code

The following code is all that is used to email out the invoices. The text is retrieved from labels that preview the email before it is sent.

```
System.Web.Mail.MailMessage email = new System.Web.Mail.MailMessage();
email.To = lblMailTo.Text.ToString().Trim();
email.From = "billing@valleysolutionsinc.com";
email.Body = lblMailMessage.Text;
email.Subject = "Valley Solutions Online Statement Notification";
email.BodyFormat = System.Web.Mail.MailFormat.Html;
System.Web.Mail.SmtpMail.SmtpServer = "192.168.0.2";
System.Web.Mail.SmtpMail.Send(email);
```

References

1. “bg2go” < http://www.bg2go.com> (19 April 2004).

2. “eDirectory – On-line Directory Software” < http://www.edirectory.com> (19 April 2004).

3. “Microsoft Corporation” < http://www.microsoft.com/> (19 April 2004).

4. “Portal Manager and On-line Directory Software” < http://www.Internetprogramming.co.uk/smart_portal_manager.html> (19 April 2004).

5. Wash, Andrew. Personal interview. 10 Apr. 2004.

6. Wong, Wylie. “Companies open wallets to portal software” < http://news.com.com/2100-1001-934828.html> (11 June 2002).