Final Design Document for
InteRx Software System

By

Smoke and Mirrors
Wade Hadlock
Justin McMurray
Jacob Tripp

CS 4500
Spring 2003
1. High-Level Description of your Software
Describe the purpose of your software and its main features (i.e., what it can do).

Our software system is a web-based database that facilitates pharmacy technicians in the compounding of chemical compounds used as dietary supplements and/or medications.

Core functionality includes:
- Checking a compound for ingredients with known harmful interactions
- Comparing ingredient quantities against recommended and maximum daily allowances
- Calculations including the total number of capsules, pills, etc. to be produced given a certain quantity of the compound to be produced
- The ability to add new ingredients into the database.

Key to the functionality of the website are pages that allow for easy online administration, i.e. adding, editing, and deleting database records.

Extended functionality includes purchase order generation; order tracking that allows both employees and customers to track the status of an order; and a database of customer information.

All of this is accessible via a secure web interface.

2. System Components
What programming language, environments, and software components (e.g., databases, OpenGL, etc.) does your system use? What platform(s) does it run on? If someone wants to use your software, are there any special system requirements (hardware and/or software) that the person's computer must have to run your software?

InteRx was built with the Microsoft VisualStudio.Net programming environment on Windows 2000 machines. The languages used were C# for the server side processing and JavaScript for the client side processing. We took advantage of the new capabilities of server side web forms to make a more intuitive interface than was previously capable via the web.

InteRx is a web application built on Microsoft’s IIS sever running on Windows 2000 Server and Windows .NET. The web server also needs to access a Microsoft
SQL 2000 Server to store and access data. In our project, the web server and database server were on the same machine, but they could also be run on separate machines.

Users of the software can access the system from anywhere on the Internet. It is compatible with Internet Explorer 5.0 and higher and Netscape 6.0 and higher, not to mention Mozilla 1.0 and up. Virtually all Internet users can access the system because those browsers run on Apple, Windows, Unix, and Linux.

3. System Capabilities
Provide a list and brief description of the (high-level) capabilities that your software can perform.

InteRx can restrict access to the system by using a secure login system. The system allows users to formulate compounds necessary for dietary supplements and calculate pricing for the order. The system saves all the data that is entered by the user, or that is calculated by the system into an SQL database. InteRx will save past orders for reference or to be modified in the future. As each ingredient is added to a formulation, the system performs error checking to make sure that no interactions are present among the ingredients, and also to ensure that the maximum dosage is not exceeded.

4. Coolness Factor
In your opinion, what is especially novel, interesting, and/or cool about your system?

The InteRx system uses new features that the ASP.NET environment provides including the use of data grids. These data grids behave similar to the way a spreadsheet program behaves while running over the web. In one .aspx page data is displayed in a table form that can be edited a row at a time. Some columns can be dropdown lists, while others can be formulas that are computing values as the user enters data.

Extensive JavaScript was used to format data fields correctly including percentages, money, and commas. The advanced JavaScript allows us to quickly format these values to store them to a database and present them correctly to the users.
The server side processing also allows a new level of form validation that makes it possible for users to see harmful interactions and identify other possible problems as the user is filling out data, and not just when it is all done being entered.

5. Individual Contributions
For each team member:
List the specific parts of the final system that were produced by each person. Any source code that did not end up in the final system should not be listed since that represents a technical, communication, or integration problem with respect to the software engineering process (you can talk about this in the lessons learned section though, if you wish).

Individual contributions:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Integration</td>
<td>Wade</td>
</tr>
<tr>
<td>Database Components:</td>
<td></td>
</tr>
<tr>
<td>Order</td>
<td>Wade</td>
</tr>
<tr>
<td>Ingredients</td>
<td>Jacob</td>
</tr>
<tr>
<td>Interactions</td>
<td>Justin</td>
</tr>
<tr>
<td>Users</td>
<td>Wade</td>
</tr>
<tr>
<td>Formulations</td>
<td>Jacob</td>
</tr>
<tr>
<td>Bottling Values</td>
<td>Justin</td>
</tr>
<tr>
<td>Capsule Filling Values</td>
<td>Justin</td>
</tr>
<tr>
<td>Table relationships</td>
<td>Wade</td>
</tr>
<tr>
<td>Visual Design and Web Interface</td>
<td>Justin</td>
</tr>
<tr>
<td>Website Configuration</td>
<td>Jacob</td>
</tr>
<tr>
<td>Restricted Access/Security</td>
<td>Wade</td>
</tr>
<tr>
<td>Web Menu Bar</td>
<td>Justin</td>
</tr>
<tr>
<td>Data Grid formatting</td>
<td>Justin</td>
</tr>
<tr>
<td>Data Grid behavior</td>
<td>Jacob</td>
</tr>
<tr>
<td>Data Mining Capability</td>
<td>Justin</td>
</tr>
<tr>
<td>Compound Formulation Checker</td>
<td>Jacob</td>
</tr>
<tr>
<td>Price Calculator</td>
<td>Jacob</td>
</tr>
<tr>
<td>Quote Sheet Generator</td>
<td>Wade</td>
</tr>
<tr>
<td>Customer contact information</td>
<td>Wade</td>
</tr>
</tbody>
</table>
Team time contribution:

- Jacob Tripp 1/3
- Justin McMurray 1/3
- Wade Hadlock 1/3

6. Accomplishments

Are you happy with your final project? Did you get everything accomplished that you wanted to?

Our team is happy with what we’ve done with our final project. We gained experience working as a team, and working for a real-life business. We are also happy that we had the opportunity to learn how to program with Visual Studio .NET and with C#. Of course we came across many ideas to enhance and improve our web pages, as is the case with most web applications. We were able to accomplish everything that we wanted to in this project.

7. Lessons Learned: Working as a Team

Discuss any lessons that you've learned from your CS4500 project about working as a team.

Teamwork in this project showed us the many benefits of working in a team, but also presented a few challenges. Our different backgrounds made it possible for us to learn from each other and made it so we didn’t spend too much time stuck on any one problem. For the more difficult code we programmed in pairs, which made it flow better. Occasionally we found situations in which one person needed to wait for the code of another to continue at a good pace on the project.

8. Lessons Learned: Building a Large Software System

Discuss any lessons that you've learned from your CS4500 project about building a large software system.

We learned a few lessons about building large software. First, we learned that careful planning must take place so that each piece of the system is done in proper order. If you’re not careful you can find yourself getting stuck needing a component of the system that hasn’t been built yet. At one point we even found ourselves wanting to modify a database table that could have been easily changed in the beginning.
We also found that is usually seemed to take longer then we thought to program a specific page or component. It seems it is always necessary in large software to schedule in time for bad estimates. It also came clear that more we communicated about what each person was doing; the more things seemed to fit together without problems.

9. Lessons Learned: Time Machine
If a time machine could transport you back to January, what would you do differently in designing, implementing, or managing your project?

Our group learned a lot about what it takes to setup a development environment that can be accessed at home and school and still communicate with a production server at a third location. We chose to work on a project for a local company to get experience with real world applications and a real world cooperation. They gave us root access to a development server that we could setup, but resided at their business. Very quickly we saw limitations due to security rights and firewalls.

Our first difficulty was with Visual Studio .NET and our web server. Our school lab does not allow us to install the IIS web server because they are afraid of possible virus attacks because of its security flaws. This is a problem because Visual Studio .NET does not allow you to start a new ASP .NET project without having access to the web server. Our web server needed to be the production server they provided because we couldn’t use any school computers. We tried to access the server with ftp, but Visual Studio wouldn’t let us. We were finally able to connect with FrontPage server extensions.

Our next challenge was that in order to use the data design tools with Visual Studio, Visual Studio needed to be able to see the SQL Server. This did not work because the school does not allow SQL traffic to get in or out of its routers. The web server could see the SQL server because they are on the same machine, but outside of their office, Visual Studio could not see the SQL server. The school did let us setup a SQL server, but they wouldn’t let up put IIS on it. Thus, the computer that they set up for us ended up being completely useless.

Finally we were able to install Visual Studio on the production server and connect via remote login to use Visual Studio’s tools.