2009

Event based propagation approach to constraint configuration problems

Rajhdeep Jandir
San Jose State University

EVENT BASED PROPAGATION APPROACH TO CONSTRAINT CONFIGURATION PROBLEMS

A Thesis

Presented to

The Faculty of the Department of Computer Science

San Jòse State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Rajhdeep Jandir

May 2009

SAN JOSÉ STATE UNIVERSITY

The Undersigned Thesis Committee Approves the Thesis Titled

EVENT BASED PROPAGATION APPROACH TO CONSTRAINT CONFIGURATION PROBLEMS

by

Rajhdeep Jandir

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

[illegible] 4/2/09

Dr. Chris Tseng, Department of Computer Science Date

[illegible] April 2, 2009

Dr. Sami Khuri, Department of Computer Science Date

[illegible] 4/2/09

Mr. Zaki Hasan, Kaiser Permanente Date

APPROVED FOR THE UNIVERSITY

[illegible] 5/19/09

Associate Dean Office of Graduate Studies and Research Date

ABSTRACT

EVENT BASED PROPOGATION APPROACH TO CONSTRAINT CONFIGURATION PROBLEMS

by Rajhdeep Jandir

The purpose of this project is to improve the search experience based on constraint problems. A real application has been developed to show the power of Asynchronous JavaScript and XML, AJAX, in web applications. AJAX allows Extensible Markup Language, XML, files to act as storage centers, thus replacing databases. This reduces latency and the amount of time needed to render the results. The design of the XML files drives the efficiency of the constraint algorithms built in the server-side scripts. The application is designed to provide the feeling of customization. The resulting information from the application is more user-specific than the information from the traditional way of designing web applications because it provides a reasonable result of the specified search to the user. A survey was conducted to reinforce that the created application is indeed user friendly since it is easy to learn and provides better results than older applications.

ACKNOWLEDGEMENTS

I would like to express my appreciation to Dr. Chris Tseng who provided the motivation, resources, and invaluable insights without which this project could never have been accomplished.

In addition, I would like to express my gratitude to my committee members, Dr. Sami Khuri, and Zaki Hasan for their constructive comments on my work.

Finally, I would like to thank my parents, sister, and brother for their continuous encouragement and support.

TABLE OF CONTENTS

Table of Contents (cont'd)

Table of Contents (cont'd)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (cont'd)

LIST OF FIGURES (cont'd)

Introduction

Turning on a computer to check mail, chat with family and friends, search information, or even make customized products online is a very ordinary activity that people perform today. Sparked by the USSR's launch of Sputnik, the Internet has come very far from its beginnings. The fear of being in a global thermonuclear war is the reason the United States wanted to research the possibility of securing the computer system so that the military could safely exchange data (Goodman). The research began in 1958 with the creation of the Advanced Research Projects Agency (ARPA) and it was not until 10 years later that the first network was created between University of California Los Angeles and the Stanford Research Institute (SRI) in Menlo Park, California. With time, more and more universities began connecting to the network allowing researchers to easily share information. The increase of technology in this area further solidified the inability to disable the network. In the 1980s, the Internet grew larger but it was still not publicly available and only used by professors for emailing purposes. It was not until a decade later that the Internet went public ("Internet," Wikipedia: The Free Encyclopedia). For a more detailed timeline of the birth of the Internet, visit http://www.cbc.ca/news/background/internet/.

From 50 million users in 1998 to 1.3 billion users today, the Internet has come a long way (Schmieg). The first activity on the Internet was email. Rather than making a phone call, sending basic text email to one another was a great advantage to researchers. With passing time, more features have been added to email such as the ability to send emails to more than one email address at a time, the attachment of files such as text files or music files, and the ability to change the text style to give it a more personal touch. This in turn triggered file sharing. Napster, a music file sharing site, was one of the first in the peer to peer (P2P) file sharing scene. P2P, however, brings in the debate of freedom versus the law. The progress of the Internet also brought in an idea that would not have been possible otherwise: the ability of engineers from all over the world to collaborate together and share ideas. This has resulted in the free software movement. Instant messaging systems have also been created to replace sending emails if instantaneous response is required. Lately, watching media online has changed the way people view their favorite television shows. Despite the entrance of TiVo, a digital video recorder that records one's favorite television shows while one is running errands, many people have resorted to watching their favorite television shows and movies online. Webcasting, which is broadcasting over the

Internet, and podcasting, which is syndicated downloading of visual/sound files, are the new norm. The Internet has also allowed remote access, commonly used in the workplace so that an employee can access the network of the organization he or she works for from any place outside. Remote access is usually achieved with the presence of security and authentication applications. The Internet has also allowed the users to have a personal connection to the Internet.

The emergence of social networks has been an area that is slowly creating waves. Despite the fact that MySpace was one of the first social networking sites, it was not until Facebook that the social network domain was revolutionized. In the latter site, people are comfortable enough to display personal information about themselves such as their real name, birthday, and even their address. Furthermore, Internet users are writing blogs, websites maintained by one or more persons with entries about a particular subject or acting as an online diary. One can also use Internet telephony called VoIP, Voice over Internet Protocol, as a means to keep in touch with loved ones living far away. The latest trend is to use the mobile phone to access the Internet. This innovation is, in a way, the world in the palm of one's hand. Research has stated that the mobile phone will be the primary device used to connect individuals to the Internet in the near future. At the same time, one important change is happening in the world as we speak. The industry is shifting gears from mass production to mass customization.

Whether one is making products to buy online or searching a website, the experience is focused to make the user feel as if it were designed for him or her. It was as early as 1982 when the XCON system was used to configure computer systems. Although the system did pose many problems, it has been enhanced since. The first company to emerge with the computer configuration idea successfully is Dell. It is the foremost company to successfully use a configuration system for the masses. Earlier when one bought a computer, he had to buy it as is. If one wanted more features, he bought those accessories and added them onto the computer himself. Dell revolutionized this idea and allowed the users to configure their own computers. It used technology that would produce quick results for the users to choose from. Its competitors, Hewlett-Packard and Lenovo, have also followed suit. Other industries like luxury car makers such as BMW and Lexus also allow users to design their own cars. The customization has not stopped at product configuration. It has also invaded the search scene. When one goes on a particular site, for example

Amazon, it creates a custom experience by telling the user what other items he may like according to the current search. Other search sites allow one to try to find something rather than to buy an item. These sites provide the capability to be able to search for certain constraints of an item. Users today go to websites that provide the customization feel with fast results. The custom configuration creates a constraint satisfaction problem (CSP) for the developers to solve. The constraint satisfaction problem involves all the variables of an item and how they are related to each other. These relations in turn produce the results the user sees.

The next section, CSP Introduction, gives some background of what the constraint satisfaction problem is and how it is currently solved. The following section, Section 3, gives the problem description to be solved followed by a discussion of the data model in Section 4. The User Interface design is discussed in Section 5 and the WAMP and AJAX tools in Section 6. Section 7 will discuss the importance of XML. Section 8 will describe the languages used in the implementation of the constraint configuration web application and the implementation will be discussed in section 9. A usability study has been conducted on the application created which will be presented in Section 10, followed by the future work and conclusion section.

CSP Introduction

As one can infer by its name, constraint satisfaction problems deal with constraints. There are many constraints in the real world that all of us deal with on a day-to-day basis. Managing home and work life or even one's monthly budget is a constraint. These constraints can be dealt with, but when there are problems consisting of a large number of variables and constraints for them, that is when computers need to come in.

Constraint Satisfaction Problems fall under the artificial intelligence (AI) realm. As stated by Eugene C. Freuder and Alan K. Macksworth, "Constraint satisfaction, in its basic form, involves finding a value for each one of a set of problem variables where constraints specify that some subsets of values cannot be used together" (13). A basic example of this is making a bike. There are many things to consider, but the constraints between the parts are not as complex and so assembling a bike is not as difficult. What is unique about constraint problems is that they follow a basic structure unlike its fellow AI problems.

Equation Format

The basic format of a constraint satisfaction problem is in three parts: the variables, the variables' respective domains, and the constraints of those variables. Let X be the set of variables, D be the respected domains associated with them and C be the constraints. Mathematically, these three would be represented as follows:

$X := x_1, x_2, x_3, \ldots, x_n$

$D := D_1, D_2, D_3, \ldots, D_n$

C := a pair consisting of (t, R), t := tuple of variables, R := variables relation with each other.

For example $x1 + x2 > 10$ is a relation consisting of two variables.

"An evaluation of the variables is a function from variables to domains, $v : X \rightarrow D$. Such an evaluation satisfies a constraint $\langle (x_1, \ldots, x_n), R \rangle$ if $(v(x_1), \ldots, v(x_n)) \in R$. A solution is an evaluation that satisfies all constraints" ("Constraint Satisfaction Problem," Wikipedia: The Free Encyclopedia). The next subsection gives examples of the areas CSP can be applied to.

Examples

SEND + MORE = MONEY

A famous CSP problem is the set of cryptarithmetic problems. In these types of problems, the digits are replaced by letters of the alphabet and the sum should also result in a correct alphabetic word, as for example, the following problem:

$$\begin{array}{r} \text{SEND} \\ \underline{+\quad \text{MORE}} \\ \text{MONEY} \end{array}$$

The variables are S, E, N, D, M, O, R, and Y. Taking into consideration that S and M are the leading letters, the domain for these variables will be from [1-9] while the remaining letters domains will be from [0-9]. Now to construct the constraints, the following equations are taken into consideration:

$$\begin{array}{r} 1000 \cdot S + 100 \cdot E + 10 \cdot N + D \\ \underline{+ \qquad 1000 \cdot M + 100 \cdot O + 10 \cdot R + E} \\ 10000 \cdot M + 1000 \cdot O + 100 \cdot N + 10 \cdot E + Y \end{array}$$

In addition to these constraints, the other restrictions are $x \neq y$, where x and y are the variables in the set {S, E, N, D, M, O, R, Y}. Solving the equations above, one gets a unique equation which looks like the following:

$$\begin{array}{r} 9567 \\ \underline{+\ 1085} \\ 10652 \end{array}$$

Zebra Example

To see an even more complicated example, take a look at the Zebra puzzle by the famous Lewis Carroll. The first five statements are the base facts followed by more detailed information.

1. A small street is composed of five colored houses.
2. Five men of different nationalities live in these five houses.
3. Each man has a different profession.
4. Each man likes a different drink.
5. Each man has a different pet animal.
6. The Englishman lives in the red house.
7. The Spaniard has a dog.
8. The Japanese is a painter.
9. The Italian drinks tea.
10. The Norwegian lives in the first house on the left.

11. The owner of the green house drinks coffee.

12. The green house is on the right of the white house.

13. The sculptor breeds snails.

14. The diplomat lives in the yellow house.

15. They drink milk in the middle house.

16. The Norwegian lives next door to the blue house.

17. The violinist drinks fruit juice.

18. The fox is in the house next to the doctor's.

19. The horse is in the house next to the diplomat's.

Now the question that needs to be answered from the above information is who has the zebra and who drinks the water. The first step is identifying the variables and its domain. With the first five statements, it can be seen that there are five variables each with five values in its domain as seen below:

- men : Englishman, Spaniard, Japanese, Italian, Norwegian
- profession : painter, sculptor, diplomat, violinist, doctor
- drink : tea, coffee, milk, juice, <water>
- pet animal : dog, snail, fox, horse, <zebra>
- color : red, green, white, yellow, blue

Despite the fact that the information does not tell what the fifth value of those two domains, drink and pet animal, is, from the question, the two unknown values can be filled to water and zebra. We convert the letters to integers and make the domain for each of the variable to [1-5]. We now go back to the statements 6-19 and assign the following constraints seen in table 1.

Table 1. Zebra Puzzle Constraints

Statements	Constraints
6. The Englishman lives in the red house.	Englishman = red
7. The Spaniard has a dog.	Spaniard = dog
8. The Japanese is a painter.	Japanese = painter
9. The Italian drinks tea	Italian = tea
10. The Norwegian lives in the first house on the left.	Norwegian = 1
11. The owner of the green house drinks coffee	green = coffee
12. The green house is on the right of the white house.	green = white + 1
13. The sculptor breeds snails.	sculptor = snails
14. The diplomat lives in the yellow house.	diplomat = yellow
15. They drink milk in the middle house.	milk = 3
16. The Norwegian lives next door to the blue house.	$\mid$ Norwegian – blue $\mid$ = 1
17. The violinist drinks fruit juice	Violinist = fruit
18. The fox is in the house next to the doctor's.	$\mid$ fox – doctor $\mid$ = 1
19. The horse is in the house next to the diplomat's.	$\mid$ horse – diplomat $\mid$ = 1

In addition to these constraints, the restrictions of each variable being unique need to be added. For example, Englishman ≠ Spaniard, painter ≠ sculptor, tea ≠ coffee, dog ≠ snail, red ≠ white, etc. From these constraints, one unique solution is obtained as shown:

- Englishman = 3, Spaniard = 4, Japanese = 5, Italian = 2, Norwegian = 1
- Tea = 2, Coffee = 5, Milk = 3, Juice = 4, Water = 1
- Red = 3, Green = 5, White = 4, Yellow = 1, Blue = 2
- Painter = 5, Sculptor = 3, Diplomat = 1, Violinist = 4, Doctor = 2
- Dog = 4, Snail = 3, Fox = 1, Horse = 2, Zebra = 5

Seeing that the Englishman drinks milk, lives in the red house, is a sculptor and breeds snails, the Japanese owns the zebra, and the Norwegian drinks water. These results can be checked against the statements and it can be seen that the solution is in fact correct. The next section provides some methods to find solutions to CSP.

Methods for Solving CSP

The above examples show how constraint problems of different areas can all be solved when they are conformed to the constraint satisfaction problem standard. There are three main methods used for solving constraint satisfaction problems, backtracking, constraint propagation, and local search.

Backtracking is a recursive algorithm which starts at the root of the tree and descends down one level at a time. This keeps going until a leaf is found. Once this happens, the algorithm goes back to the parent node and the next descendant, if it exists, is selected. The algorithm keeps doing this until the control is back to the root node of the tree and all the children have been visited. The leaves of a tree are either solved or failed. Leaves that yield one solution usually have domains that are singleton sets (Apt 65). What is unique about the backtracking method is that if only one solution is desired, the program stops after the first leaf generates a CSP solution. If all the possible solutions are desired, then the program traverses through the whole tree.

Constraint propagation is when the CSP is replaced with a CSP that is smaller yet still equivalent. This idea allows for a solution that is found in less time. The methods used in this type of solving enforce local consistency, "conditions related to the consistency of a group of variables and/or constraints" ("Constraint Satisfaction Problem," Wikipedia: The Free Encyclopedia). These problems are also used to prove that a CSP is either satisfiable or unsatisfiable. Some popularly used local consistencies are arc consistency, hyper-arc consistency, and path consistency.

Local search techniques are not favored due to the fact that they are incomplete satisfiabilty algorithms. These types of algorithms can yield to one of the two possibilities: find a solution to a CSP or do not find a solution even if one exists. This type of technique has led to hybrid algorithms, since the integration of search with local search are used together to increase the probability of finding a solution. The vertex problem, traveling salesperson problem, and Boolean satisfiabilty problem are a few problems where local search has been used.

The above three methods are used in constraint satisfaction problems. Things change a bit when the configuration becomes a part of the equation. As the problem solved in this paper falls under this category, it is beneficial to see what else is added or taken away when solving such kinds of problems. According to Daniel Sabin, configuration contains two key features that distinguish it from other types of constraint problems: "The artifact being configured is assembled from instances of a fixed set of well defined component types, and components interact with each other in predefined ways" (43). He also goes on and defines the existing paradigms of solving such problems.

The first paradigm uses rule-based reasoning. XCON is a great example of this type of paradigm. It used rules to represent domain knowledge and control strategy. XCON was the first successful configurator that used this technique for its predecessors failed using traditional programming languages. As one can guess, being consequential rules they are in the if-then form. What happens in such a system is that at each step, all the rules are examined and only the ones that fit the criteria are executed. Something to be noted is that despite several rules capable of being executed, only one rule is chosen to be carried out. One obvious problem this type of system faces is maintenance. It is extremely hard to update all the rules if a component specification has changed. This in turn gives through to the model-based reasoning.

Model-based reasoning bases its assumption that a model exists which contains "decomposable entities and interactions between their elements" (Sabin 44). Sabin also states that model-based reasoning provides "a better separation between what is known and how the knowledge is used, enhance[s] robustness, enhance[s] compositionality, and enhance[s] reusability" (44). Robustness is increased because the model allows for a broader range of problems. Compositionality, because of the robustness, allows the ability to combine different domain knowledge into one model. As a result, the model is reusable to solve related problems.

The third technique Sabin discusses is case-based reasoning. This approach uses the old configurations that have been made by users. New configurations are made by looking at a previously similar problem and using the solution it provided. That solution is used and modified to fit the current configuration. The new configuration is also stored for future reference. This technique involves four basic steps. The first step is the input for the configuration provided by the customer. The next step is finding a configuration that is most similar to the user-specified requirements. The following step is to adapt the current configuration to the existing solution. The last step is to store the new configuration for future reference. One advantage this technique has over its counterparts is that a configuration can be made even if all the requirements have not been given. This is because a configuration that closely matches the set of given requirements is retrieved and unlike rule-based and model-based reasoning, a customer is able to find a product when he has not provided all the requirements. With the increasing trend of the Internet moving towards user customization over mass production, J. Toussaint and K. Cheng have introduced web-based reasoning.

Web-based reasoning uses the Internet as a component in its system. It is an approach seen to help knowledge-based systems. Toussaint and Cheng state that competition has led to researching in an Internet-based solution to increase the efficiency of the processes of their product, tool configuration (24). This web-based system uses case-based reasoning (CBR) for choosing the best tools in a configuration. Of course, initial parameters need to be given to the system so that the choice of tool can be made. A feature Toussaint and Cheng have added is "a user-driven 3D turning simulator which allows testing the chosen insert for several turning operations" (24). The application's main purpose is an e-manufacturing tool that provides tooling data quickly with the utilization of user interaction. In their configuration, Toussaint and Cheng add the human factor which has been missing in earlier versions and incorporate CBR, so that the configurations that engineers make can be stored for later use. The paper shows how industries are also changing their development process to provide a more customized feel for their customers. The next section will explain how these models are programmed.

Constraint Programming

Constraint Programming is known as the process of transforming constraint satisfaction problems. They are transformed by first defining a set of variables, which are a specific range. Then the constraints are created keeping the range in mind and a language is chosen to represent the constraints. Constraint programming is a programming paradigm where relations between variables can be stated in the form of constraints. Constraints differ from the common primitives of other programming languages in that they do not specify a step or sequence of steps to execute but rather the properties of a solution to be found. As mentioned earlier, it is important to maintain equivalence in the transformed and original CSP so that the right solution is generated. The first phase involves modeling, which includes the variables, domains and constraints, and the second phase is the problem solving. The CSP model needs to be perfect so that the correct solution is reached. If it is not modeled correctly, then the CSP may or may not yield the right answer (Apt 58-9).

Prolog uses constraint logic programming. The purpose of constraint programming "is to search for a state of the world in which a large number of constraints are satisfied at the same time" ("Constraint Programming," Wikipedia: The Free Encyclopedia, par. 6). With the increase of customization, constraint programming is gaining importance and as a result is being extensively researched. The next section provides some of the configurators that are available for use today.

Configurators Available

Because CSP has been around for quite some time and constraint configuration for only about twenty years, constraint configuration still is an area yet to be explored exclusively. With the rise of open source, there are a number of private and free configurators available for the user. ILOG is the most prominent private software that makes business rule optimization used in enterprise applications. Configurators, solvers, are now available in multiple languages. Koalog, a Commercial Java library provides results using constraint programming or local search. Complex System Technologies (COSYTEC) is a constraint programming tool based on Prolog. LINDO and LINGO solvers are both optimization solvers. With the notion of open source, there are more tools available in this area (Ceberio).

CHOCO, an open source Java constraint library, is based on constraint programming using backtrackable structures. Constraint Language in XML (CLiXML) is an open-source software application designed to constrain the content of XML documents. The constraints in CLiXML are described by either using first-order logic or XPath expressions. Cream is yet another open source Java library for constraint programming. CSolver, a constraint solver written in C, uses the arc consistency algorithm mentioned earlier. Open Tax Solver is an application written in C to fill out tax forms. Geocode is a kernel providing search and finite domain constraints implemented in the C++ library. One feature this application contains is its feasibility to be interfaced with other systems such as Java. There are many more open-source constraint libraries that programmers have built for others to use. The next section will explain the problem with current configurators.

Problem Description

General Problem

With the passing years, constraint configuration has been incorporated into web applications. It is one of the tools used to provide customization for the users. Configuration in web applications is becoming quite dominant now. It is a way to give the users control over what they want to see in the end result. The applications allow the users to refine and select categories that will make their search more relevant and have better, more useful results. As the user interacts with the categories given, the web application allows the user to select and unselect all the product constraints of that category. At the same time, as a category is selected, other categories are filtered out because of the inconsistency with the current query. This produces a distributed constraint satisfaction problem (dCSP). A solution, which provides great user experience, needs to be incorporated to solve such a problem. Constraint programming is the paradigm that is used as a tool to solve CSPs. As stated by E. Freuder, "Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it" (Constraint Solving, par. 1).

Users want immediate feedback when they are configuring their searches. Websites that take some (more than a few seconds) time to render the results are not visited again. This is because we live in a fast-paced world now where not just time is valued, but so are correct results. Websites that are not configured properly and do not yield high-quality results are also not given a second glance. As a result, many factors need to be considered when designing customized web applications today.

My Solution

The goal in solving such a problem is to incorporate new technologies while using the idea of constraint configuration as the backend of solving the problem. After much research Asynchronous JavaScript and XML (AJAX) has been chosen as the technique to quickly update the web page without having the need to refresh the whole page but only what is needed to. XML has been incorporated as the source of data holder. The need and use of a database has been done away with in this project which will be explained in the sections AJAX and XML. The application that is created uses these and other tools to create the best user experience possible with reliable results. The configurator is initiated when an event happens. That is, when the user makes an action, the event propagation begins. The selection is sent to the backend and the CSP problem is solved. This will be discussed in detail in later sections. The first step in creating an application is to have a data model.

Data Model

The Flow

With the technologies being used and a certain result desired, there is a specific flow that the application follows. The application is a user-driven event propagation system and so until the user makes a selection there is no action being performed. It is with the user defined constraints that the problem results in a solution. The main basic flow is shown in figure 1 below.

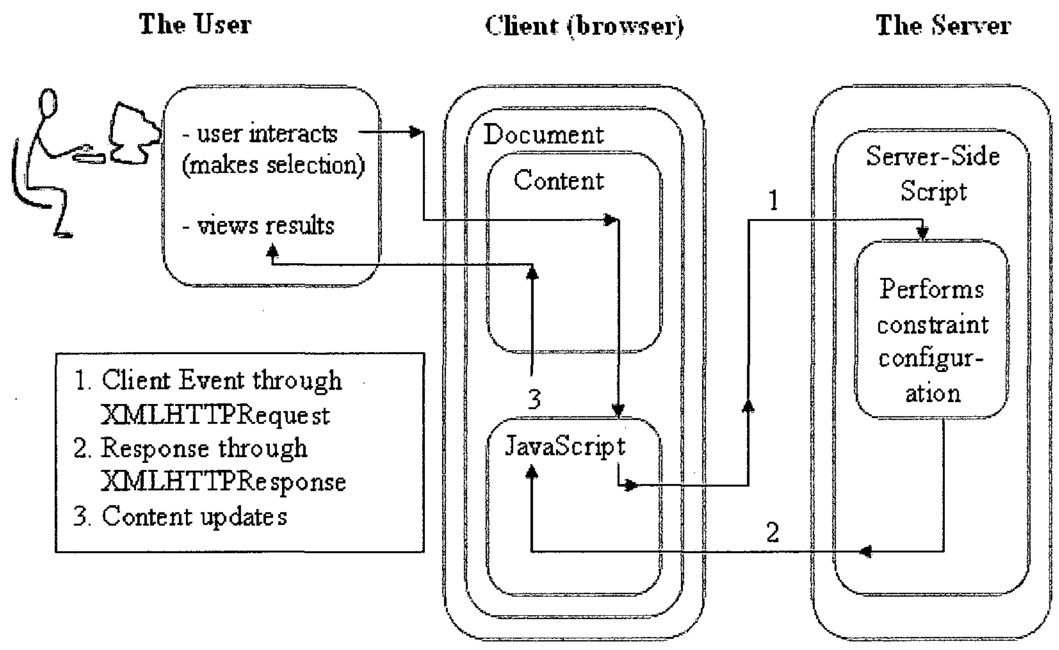

The User
Client (browser)
The Server
- user interacts (makes selection)
- views results
Document
Content
JavaScript
Server-Side Script
Performs constraint configur-ation
1
2
3
1. Client Event through XMLHTTPRequest
2. Response through XMLHTTPResponse
3. Content updates

Figure 1. The Basic Application Flow

The figure above shows the basic flow of how the web application works. The user makes a selection on the web application. This triggers an event, which goes to the JavaScript function. The JavaScript function then passes the required information to the server-side script. This script performs the required configuration algorithm and sends the respective results. The results are passed from the JavaScript function to the browser for the user to see. The details of the AJAX calls are discussed in a later section.

The Components

The solution contains three main components. The first component is the User Interface (UI). The representation of the data and how users can select the constraints is of utmost importance. It is through the UI that the constraints will be passed on to the backend to solve. If the UI is not represented properly, the results will not be correct and the user experience will be poor. The second component is tools. These tools build the web application and are also of significance. Tools represent constraints themselves in the type of procedures that can be performed with them. That is why it is necessary to first gather all the possible tools that can be used and based on their advantages and disadvantages, choose the one(s) that fit the problem description. The last component that makes a good web application is the programming languages used. Again, each language poses pros and cons and based on what is desired, the language is chosen for each need. The following sections will talk in detail about the mentioned components.

The UI

When designing the web application, there were a few factors that needed to be considered. The type of UI elements, the colors used, and the layout of the web application are the three areas of concern. It was also important to meet the usability and user experience goals of interaction design. The usability goals as stated by Jennifer Preece, Yvonne Rogers, and Helen Sharp in Introduction Design: Beyond Human-Computer Interaction, are (14-16):

- Effective to use: How good a system is at what it is supposed to do.
- Efficient to use: The way the users can carry out the tasks.
- Safe to use: Is the application safe and not dangerous.
- Has good utility: "Does the system prevent users from making mistakes. and if they do, does it permit them to recover easily?"
- Easy to learn: Is it easy to learn how to use the system.
- Easy to remember how to use: How easy is it to remember how to use the system once it has been learned.

Preece, Rogers, and Sharp also state the following user experience goals (18):

- Satisfying
- Enjoyable
- Fun
- Entertaining
- Helpful
- Motivating
- Aesthetically pleasing
- Supportive of creativity
- Rewarding
- Emotionally fulfilling

The user experience goals and usability goals do influence each other. For example, the user experience goal aesthetically pleasing connects to the usability goals of easy to use and easy to remember. It is important to keep these goals in mind so that the requirements and purpose of the application are not forgotten.

The next set of UI principles considered were interaction styles, widgets to use, and the color and text schemes to use. Choosing an interaction style did affect the type of widgets that would be used. The type of interaction style is also dependent on the application's purpose. Again, the goals mentioned above are a good reference for this.

Debbie Stone, Caroline Jarrett, Mark Woodroffe, and Shailey Minocha discuss the types of interaction styles in User Interface Design and Evaluation. They state the five different interaction styles: command line, menu selection, form-fill, direct manipulation, and anthropomorphic. Command line is using the keyboard to explicitly execute actions. A clear example of this is the command prompt. The user must type each action that he or she wants done. Menu selection allows users to choose from a set of options. The selection of an option will result in a certain feedback by the system. Form-fill interaction is when the system needs to gather much information about a user. When filling out an application online, the system uses this kind of interaction style. Direct manipulation, DM, allows the "users to interact directly with the UI objects" (Stone et all. 213). When dragging files from one folder to the other or using CAD, computer-aided design, the DM style is being used. The last interaction style, anthropomorphic, "interfaces aim to interact with users in the same way that humans interact with each other" (Stone et all 215). Applications that involve gestures, facial expressions and/or eye movements fall under this style. It is after examining each style that the decision to use a combination of the menu and form-fill interaction styles was made.

Menu selection is very easy to learn and involves very few keystrokes by the user. In research, it has been stated that users do not like performing more than a few actions before wanting to see some results. In the application, the menu also allowed grouping of the options. Constraints that fell under the same option type were grouped together. This decreased the amount of space used on the application and increased efficiency. Another advantage of menu selection is that it is great for learners and for infrequent users. This satisfies the memorability goal established earlier. The form-fill style will be used when the menu style seems unfit. When there are too many options to be displayed it will be better for the user to type in the option instead of having to scroll to find the particular option to choose. It simplifies the data entry that is required by the system. Since this is a web application the two interaction devices used are the keyboard to enter data and the mouse to select the options from the menu. Once the style has been chosen, the next step is to choose the widgets that will be used.

Having chosen menu selection and form-fill interaction styles, the kind of widgets needed are known but not the type. It is known that a menu, text area, and button are needed on the web application. The type that is going to be used in each widget is dependent on the functionality of the application. There are four types of menus: drop-down, cascading, roll-up and pop-up menus. Drop-down menus are the most common and that is the type that is used for one of the implementations. The other implementation is more advanced and therefore uses cascading menus, which allow a number of menus to be displayed.

Although the user has to select a menu to see its options, it is necessary to keep in mind not to have too many menus so that the user does not become overwhelmed with the information. The use of a text box is only advantageous when the type of input is known. This way checking can be done to make sure the correct format was inputted. A text field was chosen as the type of text box for the user to enter information. The use of checkboxes was also incorporated in the second implementation discussed in a later section. The checkboxes were used as a way for users to select the options in a menu. Since the application involves the ability of multiple selection in different menus, a checkbox is used to remind the user of selections he has already made. Command buttons were used next to text fields. When the user inputs the data, he or she clicks on the button so that the system receives the entered information. Without the use of the button, the system would not know when the user wants the system to read the entered information. After having chosen the widgets, the font and text color were next to be decided upon.

It is important to make sure that the font on the application is legible. The four things to consider in font are typeface, type size, letter spacing, and line spacing. Sans serif typefaces are recommended in which Times New Roman or Arial is preferred (Stone et all 249). Since the screen has less resolution than paper, a font size between 11 and 14 is recommended. It is also important to have the right amount of letter spacing for too little and too much spacing yields in hard to read words. Line spacing also needs to be the correct amount, for too little spacing makes it hard to read and too much spacing gives the notion that the lines may not be related. After considering these requirements, a sans serif typeface Arial font of size 12 with normal spacing was chosen. Choosing the right font color was also important.

Colors represent different feelings and it is key to make sure the wrong emotions are not aroused by the web application. After looking at the color scheme, a color combination of blue, ivory, and gray was chosen. Blue represents calmness, trustworthiness, intelligence, and control (Stone et all 253). These are the emotions required for the application. Users should feel they are in control of the configuration, yet should not get frustrated at the same time. White represents cleanliness and purity (Stone et all 253). An ivory shade was chosen as the background of the second implementation to promote the user to feel a sense of completion. When the results are displayed, the user will feel a sense of satisfaction. Gray is a neutral color blending emotions. It was a color used to tie blue and white together in the application. The use of the color gray was decided over using the color black to prevent the feeling of mystery. The feeling that something was hidden was not desired in this application. The next section delves into the tools that were used in the application.

WAMP and AJAX

WAMP

A local web server was needed to write and test the web application. Apache is one of the most widely used web servers today. WAMP, Windows Apache MySQL and PHP, is a popular package that provides all these tools together. With WAMP there was no need to install any of these tools separately. Apache, as mentioned earlier, is a web server allowing the programmer to see his or her code in a web browser. This is to let him or her see how the application would look like in the web. MySQL is a database manager where information can be stored in an organized way. "PHP is a scripting language which can manipulate information held in a database and generate web pages afresh each time an element of content is requested from a browser" ("WAMP," Wikipedia: The Free Encyclopedia, par. 2). PHP will be talked about in detail in one of the following sections.

AJAX

Based on JavaScript and HTTP requests, AJAX is used to create interactive web applications. AJAX is not a new technology, but uses existing tools in a unique way. Introduced by Google in 2005, AJAX created a new era of building faster, more user-friendly web applications. AJAX reduces bandwidth usage and loading time since the whole web page is not refreshed after a request. The web application is also more interactive and parts of the web page can be updated at different times and not all at once. It furthermore reduces connections to the server since the scripts and style sheets being used only have to be loaded once ("Ajax (programming)," Wikipedia: The Free Encyclopedia). The four basic tools it uses are JavaScript, HTML, XML, and CSS.

JavaScript can be used to communicate with the server via HTTPXMLRequest. With this innovation, the data can be exchanged and the web page can be updated without having to reload the entire page. "AJAX uses asynchronous data transfer (HTTP requests) between the browser and the web server, allowing web pages to request small bits of information from the server instead of whole pages" (Ajax Tutorial, par. 4). Internet Explorer creates ActiveXObject while Firefox supports the JavaScript HTTPXMLRequest object. Each time the JavaScript method connects to the server, an HTTPXMLRequest object needs to be created first. The current state of the HTTPXMLRequest object can be accessed through the readyState attribute. The attribute holds five states as seen in the table below.

Table 2 The number of States of a HTTPXMLRequest Object

State	**Description**
0	The request is not initialized.
1	The request is established.
2	The request has been sent.
3	The request is being processed.
4	The request is complete.

The way the output is seen on the web page is through the onreadystatechange property of the HTTPXMLRequest object, which processes the response from the server. The programmer defines a function to be executed when the response is received and provides the HTML element where the output should be seen. To send the request to the server, the open() and send() methods are used of the HTTPXMLRequest object. The open() method takes three parameters. The first parameter is GET, or POST. The second parameter is the url, path, to the server side script. The last parameter is type boolean to state if the event should be handled asynchronously or not. The send method takes in one parameter in which if GET is being used, null is passed. The server-side script performs its algorithm and returns certain data to be written. The figure below explains the flow of how AJAX works.

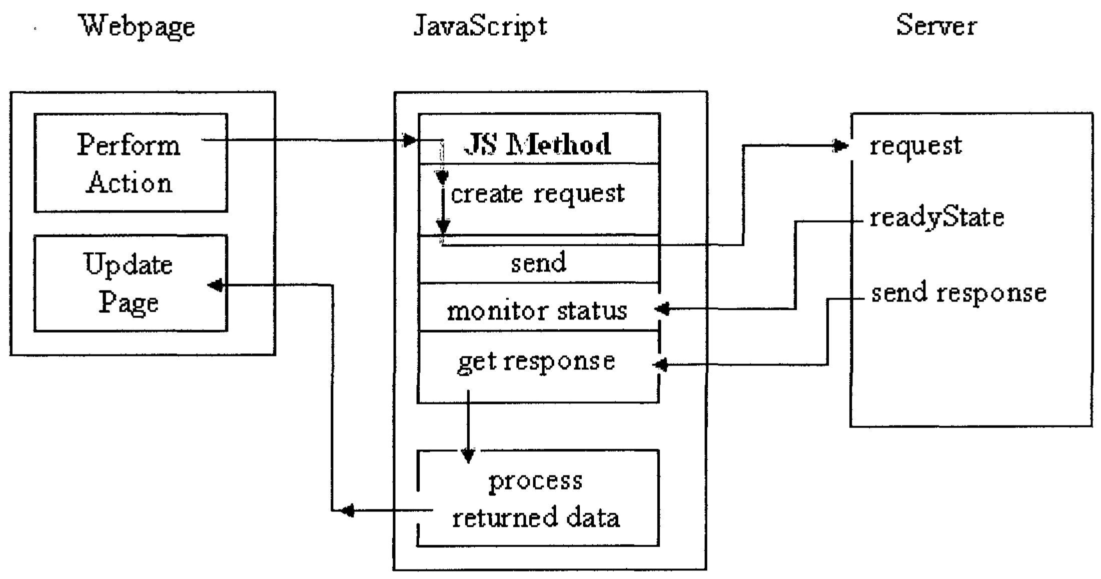
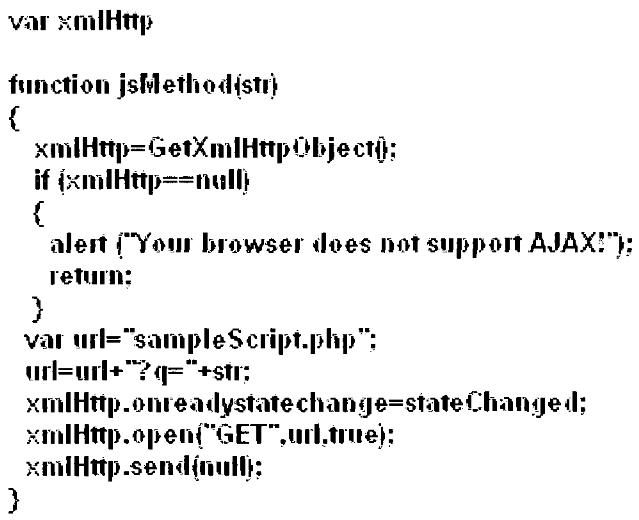


Figure 2. HTTPXMLRequest Flow

When the user performs an action, a JavaScript method is invoked. The JavaScript method can contain all the calls to create, send, and process the HTTPXMLRequest object. A sample code snippet shown below in figure 3 illustrates how this can be done. The call to the server-side script as seen in figure 2 and figure 3 is what initiates the configurator. The configurator then sends the response back to the JavaScript method which updates the web page accordingly. Figure 3 shows the code snippet of the JavaScript method creating, sending, and processing the HTTPXMLRequest object.

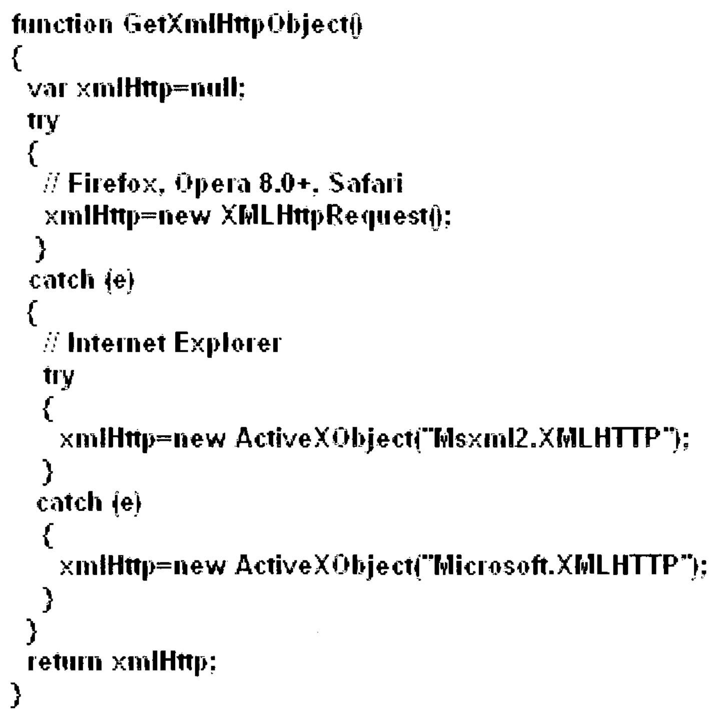


```
var xmlHttp

function jsMethod(str)
{
  xmlHttp=GetXmlHttpObject();
  if (xmlHttp==null)
  {
    alert ("Your browser does not support AJAX!");
    return;
  }
 var url="sampleScript.php";
 url=url+"?q="+str;
 xmlHttp.onreadystatechange=stateChanged;
 xmlHttp.open("GET",url,true);
 xmlHttp.send(null);
}
```

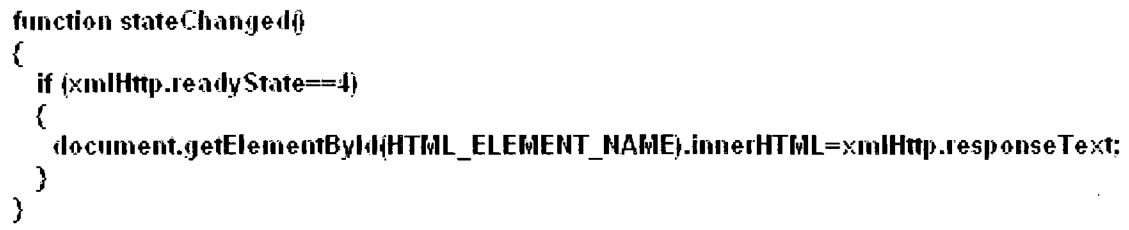
Figure 3. JavaScript Method to Initiate AJAX Mechanism

As seen in the three code snippets figures below, figures 4, 5, 6, creating an AJAX web page is quite easy. The methods stateChanged, and GetXMLHttpObject can be used by many JavaScript functions. The sever-side script shown in figure 6 just displays the variable passed in figure 3. The type of AJAX shown does not use any AJAX API that is found on the Internet today. It is the simple way of using AJAX. This type of AJAX is used in the implementation. Despite the fact that JavaScript was not supported on mobile phones, that scene is changing with more and more applications using this language and the mobile browsers are now beginning to support it. The next section states how XML has been incorporated with AJAX.



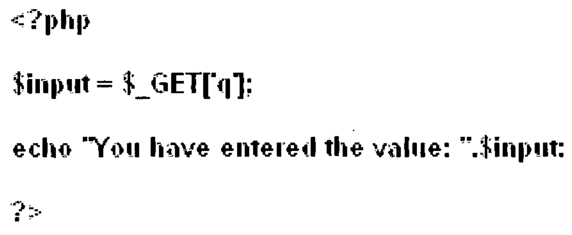
```
function GetXmlHttpObject()
{
  var xmlHttp=null;
  try
  {
    // Firefox, Opera 8.0+, Safari
    xmlHttp=new XMLHttpRequest();
  }
  catch (e)
  {
    // Internet Explorer
    try
    {
      xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (e)
    {
      xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
  }
  return xmlHttp;
}
```

Figure 4. Create XMLHttpRequest Object.



```
function stateChanged()
{
  if (xmlHttp.readyState==4)
  {
    document.getElementById(HTML_ELEMENT_NAME).innerHTML=xmlHttp.responseText;
  }
}
```

Figure 5. Update Which Part of the Web Page



```
<?php

$input = $_GET['q'];

echo "You have entered the value: ".$input;

?>
```

Figure 6. Sample Server-Side Script.

XML

Extensible Markup Language (XML) is a markup language like HTML. Its purpose, however, is different than that of HTML. Instead of displaying data, its function is to carry data. For that reason, XML does not have predefined tags like HTML does. The XML writer defines the names of the tags in accordance to the type of data that is being stored. It is a great tool that allows information sharing between computers, applications, and organizations. One point to note is that XML by itself does not do anything. It is merely storage and code must be written to access, send, or display the XML. Due to its great flexibility, XML has quickly gained popularity since its conception.

Standard General Markup Language (SGML) was invented by three IBMers in the 1970s to "describe a way of marking up technical documents with structural tags" (XML: An Introduction – Brief History, par. 1). It wasn't until a decade later in 1986 that SGML was recognized as an international standard for markup languages. It led to the creation of HTML by Tim Berners-Lee in 1991. Despite the power of HTML, its restricted tags limited the effectiveness that SGML had. Another decade later, Jon Bosak, an employee of Sun Microsystems, remodeled SGML to be just as powerful but more simple like HTML. Much work was done by Bosak and his team and the result showed when in 1998, the first XML version was recommended by W3C, World Wide Web Consortium. The accomplishment was visible with XML having just twenty-six pages of specification versus over five hundred pages of SGML (XML: An Introduction – Brief History).

What makes XML unique is the separation of HTML and XML that can be made. As aforementioned, XML stores specifically data while HTML focuses on the display of data. The HTML can access the XML data by using JavaScript in the webpage. Using databases can be tedious because computer systems and databases contain data that can many times be incompatible. With XML, this issue does not arise. It also allows data to be exchanged more easily as the incompatibility issue goes away. Because XML data is stored in text format, upgrading to new operating systems, applications, and/or browsers will not pose any hindrance in the ability to access the data in the XML. Parsing such a file is very simple and efficient. As seen, XML is independent of software, hardware, and applications making it more available and useful (XML to HTML). XML is so powerful and useful that Internet languages such as XHTML, WSDL, WML, OWL, and many more have been created with XML. It is extremely popular in the area of storing and describing information. Taking consideration of all these facts, using XML over a database to store data was decided. Choosing XML will also reduce database latency and in effect make the web application faster in returning results.

There were many challenges while designing the XML files for the application. For one, constraint configuration has not used XML to date. All the systems use algorithms that store data either in data structures or use a database. The system being designed was not using either of those. In addition, a new system that the application would use was being designed where none of the configurators available were going to be used. After much research, a paper by the Organizing Committee of Third International Competition of CSP Solvers was found titled XML Representation of Constraint Networks Format XCSP 2.1. In it, the committee designs the XML to represent input for a configuration system like CHOCO. Snippets of this XML style can be found in Appendix C. XML Spy was used to create the schemas for the XML files that are used in the application. From these schemas the XML files were also created. XML Spy allowed the ability to use XQuery on the XML files that were created to verify the logic of the XML file. XSLT was also used to create a UI look taking in the XML file as input. XML did prove very beneficial in the application. The specifics of the XML files used will be discussed in a later section, Implementation.

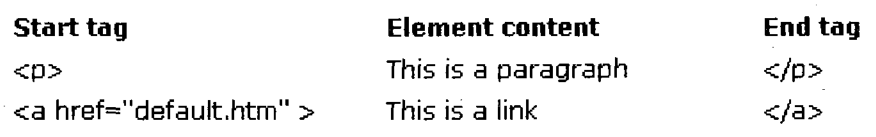
Languages Used

The application resulted in the use of many programming languages and tools. Each language and tool served a specific purpose for the totality of the system being created. In other words, each language and tool used further enhanced the system's capability and the goals. The following sub-sections discuss the reasons behind choosing each language and tool.

HTML

HyperText Markup Language (HTML) is the main language used to display the data. It consists of specific tags that tell how to display the data. The tags come in pairs like <html> </html> where the data is in between the tags. Tags such as <b>, <h1> describe the look of the text that is in between the tags. One can also create UI elements with HTML tags like drop down menus, buttons, and links. The reason HTML was chosen is because it is the language used to describe web page content. It is very simple yet powerful at the same time. It allows JavaScript to be embedded inside of it. This is a great advantage because the UI elements now have a JavaScript method with it so that it can catch the event. With JavaScript, HTML can access the contents of XML. The basic concept of HTML elements can be seen in table 3 below.

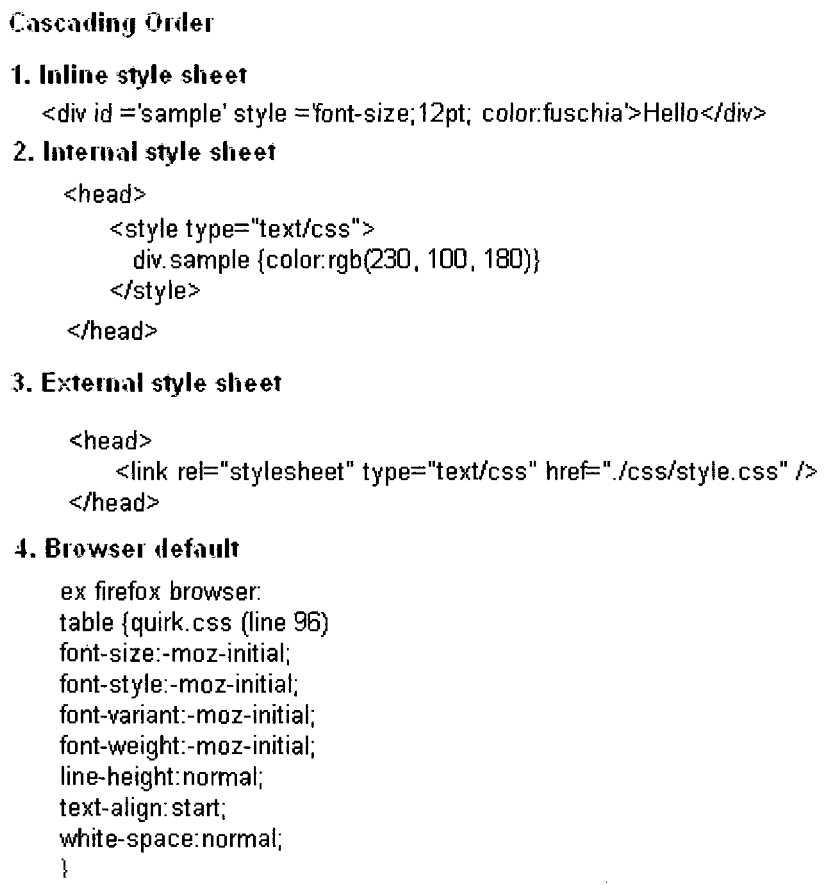
Table 3 HTML Element Syntax.



Start tag	Element content	End tag
`<p>`	This is a paragraph	`</p>`
`<a href="default.htm" >`	This is a link	`</a>`

CSS

Cascading Style Sheets (CSS) is a style language used to control the style and layout of the UI elements in a web page. It is a way to separate document content and document presentation. The purpose of style sheets is to save time and work. If there are many buttons in a web page and all have a distinct look, then it is better to define a style for them and use the name of the style in all of the button tags rather than styling each button separately with the same styling information. In CSS, if the style needs to change, only one place needs to be modified and the results will be seen on the web page. If a style sheet is not used, then each of the elements in the HTML would have to be modified. This can lead to errors as one element may be skipped. According to w3schools, the term cascading is from the expression that there can be more than one style sheet for an HTML page. The following figure shows the order of importance if more than one style sheet is present in a HTML page. The first point has the highest priority and so an inline style sheet will override and internal style sheet. However, if the external style sheet is declared in the head tag like shown in figure 7, then it will override the internal style sheet. Examples of each type of style sheet are given in the figure.



Cascading Order

1. Inline style sheet

```
<div id ='sample' style ='font-size;12pt; color:fuschia'>Hello</div>
```

2. Internal style sheet

```
<head>
     <style type="text/css">
       div.sample {color:rgb(230, 100, 180)}
     </style>
</head>
```

3. External style sheet

```
<head>
     <link rel="stylesheet" type="text/css" href="./css/style.css" />
</head>
```

4. Browser default

```
ex firefox browser:
table {quirk.css (line 96)
font-size:-moz-initial;
font-style:-moz-initial;
font-variant:-moz-initial;
font-weight:-moz-initial;
line-height:normal;
text-align:start;
white-space:normal;
}
```

Figure 7. Cascading Style Sheet Order

There are many advantages in using CSS. It allows flexibility in how the look of an element should be. Each application has a specific goal and audience and the look of UI elements on a page helps promote that. It also gives the web application unity and conformity. If the application consists of many web pages, then using the same style for all the same UI elements will tie the pages together and make it look as one. The style sheet also gets stored in the browser's cache so that it does not need to be reloaded. Therefore, it reduces data transfer. The HTML file contains only one line telling it which CSS file to use. Changing just that line to refer to another CSS file will dramatically change the look of the web application. In the web application that is designed, the style sheet is used for the background, font format, and results format.

JavaScript

JavaScript is a scripting language for the web. It is used on the client side to add functionality to the UI elements. It is extremely lightweight. The script can be embedded inside of the HTML page or in a js file which the HTML page includes. Another great advantage of using JavaScript is that it does not need to be compiled. Despite the similarity in name, it is not related to Sun's Java. One is able to write text into an HTML page with the function document.write(). It can also read the content or even write the content of a UI element on the HTML page. For example, retrieving the content of a text field element, and then setting its value to an empty string can be done using JavaScript. Through JavaScript, the type of browser the user is using can be detected. This is extremely useful when creating the XMLHttpRequest objects discussed earlier. Another feature that JavaScript has is it contains the functions that capture the user's event. When a selection is made by the user, a JavaScript function is called upon which starts the propagation as seen in figure 1. The web application created uses all these functionalities that JavaScript offers.

PHP

PHP is a powerful server-side scripting language used for web development. Originally meaning Personal Home Page, it now stands for Hypertext Preprocessor. Although the application created does not use a database, PHP supports many databases. It runs on many platforms, operating systems and is compatible with most web servers. Text can be outputted with the methods print and echo. The PHP code is compiled at runtime increasing the execution time but PHP compilers and PHP accelerators exist to overcome this hindrance. PHP also offers unique XML parsers.

PHP mainly uses two types of XML parsers, DOM and SimpleXML. DOM is a tree-based parser where it provides access to the elements in terms of tree elements. To access a certain element level, all the elements in that level must be looped through to get the desired one. SimpleXML, on the other hand, does not require the programmer to loop through all the elements in order to access a particular one. This is done through the use of XPath. XPath allows the programmer to provide the path to the element that is desired. It returns the result in an array that can be traversed through. Take the following XML document seen below to illustrate the key difference between the two parsers.

```
<books>
        <!-- This is a comment -->
        <book id="1" isbn="3-8266-0612-4">
                <title>Apache Web-Server</title>
                <year>2000</year>
                <dc:subject>Webserver</dc:subject>
        </book>
        <book id="2" isbn="3-8266-0550-0">
                <title>Linux für Internet und Intranet</title>
                <year>2000</year>
                <dc:subject>Operating Systems</dc:subject>
        </book>
        <?php print "this is a processing instruction"; ?>
</books>
```

This is an XML file containing information about books. If the book with the name Apache Web Server was desired, then the following loop is how the result would be retrieved in DOM:

```
foreach($xml->book as $book)
{
        if($book->title == 'Apache Web Server')
         .... perform actions on the object here and break;
}
```

The variable $xml is the pointer to the XML file that has been loaded. In this example it is the XML document displayed above. In SimpleXML the following would be written:

```
$result = $xml->xpath("/books/book[title = 'Apache Web Server']");
```

As can be seen, SimpleXML can achieve the same results in one line of code. It may not be easy to see with the given XML file, but for large XML files with hundreds of elements, looping through all the elements when they are not needed can be unnecessary work that the PHP script is performing. For this reason, the use of SimpleXML was chosen over DOM. Also, there is a growing liking and use of SimpleXML. As a result, more features are being added to this parser.

To get the information sent by the XMLHttpRequest object, the $_GET and $_POST variables are used. Depending on the method that was used to send the request, the programmer uses the corresponding variable in PHP. The input of either variable is the name of the variable that was passed. For example, from figure 3 to get the str value that was passed from the JavaScript function, in PHP the following line of code would be used: *$input = $_GET['q'];*. This gets the value of str which was passed with the name q and stores it in a local variable called input. In the PHP script the input variable will be used to access the content passed.

Yelp API

For one of the applications created, the use of data was acquired by using the Yelp API. The implementation involves restaurants that are found in each city. To get the restaurants from each city, the Yelp API was needed. A key was created that was used when accessing all the restaurants of a particular city. The information extracted was stored in an XML file. Details about the XML and its use will be discussed in the next section, Implementation.

Implementation

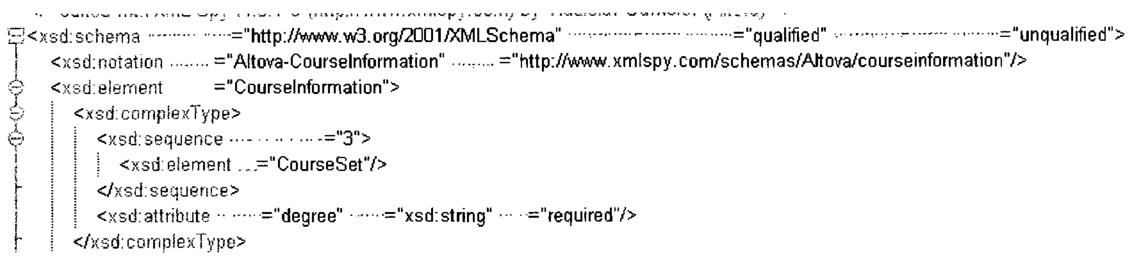
The implementation was necessary to illustrate the event based propagation approach theory on solving constraint configuration problems. The idea is very basic. Until a user makes some sort of selection, the configurator in the backend is not invoked. When a choice is made on the web application through the use of the HTML elements present, an event happens where a corresponding JavaScript method is initiated. This method then performs the AJAX steps and calls the necessary PHP file. The PHP file is the configurator which gets the user input and performs the algorithm. It then returns the result. The event based propagation approach is applied to web applications where items are searched through different constraints or products are configured.

Two applications were developed in this project. The first one was very basic and was created to see the theory work with a real world application. The second application is much more complex. The first sub section will discuss the first application followed by the second one.

Courses

Till now the problem description and the tools used for the application have been discussed, but not the implementation. Before beginning to implement a complex application, it was important to start with a smaller much simpler one to understand the nuances that will be needed to be focused upon. Being a part of the Computer Science Department at San José State University, it was decided to work on an application of course pre-requisites. This is a CSP problem, yet is simple enough to implement so the relationship between tools and languages being used can be understood.

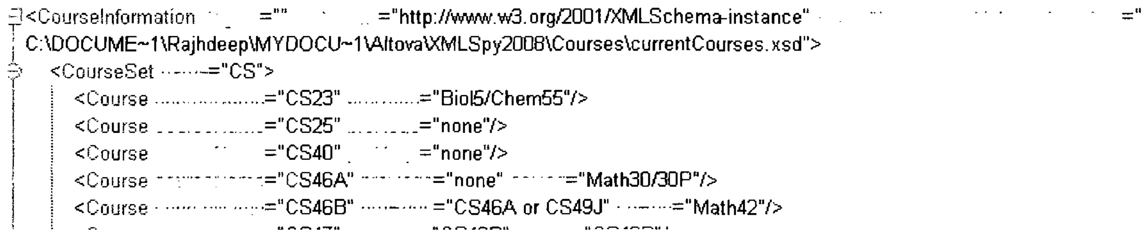
The first step was to attain the data. The courses in the Computer Science department were attained from the department website. Originally, only the undergraduate courses were considered, but later on, the graduate courses were added as well. The information needed from each course was its title, co-requisites, and pre-requisites. This information needed to be stored in the XML. The key point is how to store the information. The formatting of the XML was crucial. This is because the constraint algorithm implemented will use the XML design. The XML schema went through many revisions until it was finally agreed upon. The next step was creating the XML file using the schema defined. This was to verify that the XML file was in the correct format. It was decided that since there are three types of courses in the Computer Science (CS) curriculum, the courses would be stored in that way also. The three sets are CS courses, math courses, and General Education (GE) courses. The math and GE courses are in accordance with the courses the CS department provides. They are not all the math and GE courses that San José State University, SJSU, provides. Below in figure 8, a snippet of the schema can be seen. Appendix C offers the whole schema.



```
<xsd:schema ="http://www.w3.org/2001/XMLSchema" ="qualified" ="unqualified">
    <xsd:notation ="Altova-CourseInformation" ="http://www.xmlspy.com/schemas/Altova/courseinformation"/>
    <xsd:element ="CourseInformation">
        <xsd:complexType>
            <xsd:sequence ="3">
                <xsd:element ="CourseSet"/>
            </xsd:sequence>
            <xsd:attribute ="degree" ="xsd:string" ="required"/>
        </xsd:complexType>
```

Figure 8. Course Schema

The figure above shows the three main elements that are in the XML. As mentioned, there are three sets and that is visible by the line that says there will be a maximum of three elements with the name CourseSet. Each CourseSet will contain courses. With this schema, the figure below, figure 9, shows a snippet of the XML file that is created.



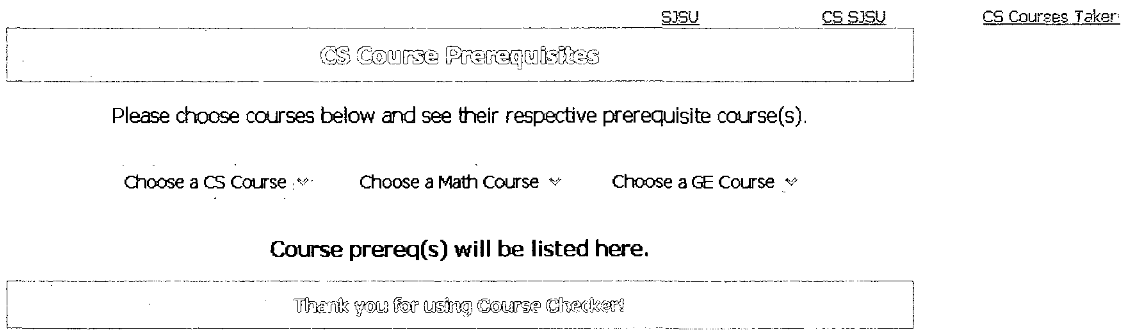
```
<CourseInformation ="" ="http://www.w3.org/2001/XMLSchema-instance" ="
C:\DOCUME~1\Rajhdeep\MYDOCU~1\Altova\XMLSpy2008\Courses\currentCourses.xsd">
    <CourseSet ="CS">
        <Course ="CS23" ="Biol5/Chem55"/>
        <Course ="CS25" ="none"/>
        <Course ="CS40" ="none"/>
        <Course ="CS46A" ="none" ="Math30/30P"/>
        <Course ="CS46B" ="CS46A or CS49J" ="Math42"/>
```

Figure 9. Course XML

In figure 9, the CourseSet with the name CS can be seen. A few of the courses that belong to this set are also visible. Each course contains the name, its pre-requisites, and co-requisites. The first two are required attributes of each course while the last one is optional because not all courses contain co-requisites. With the XML, an XSL, Extensible Stylesheet Language, was created to display the contents of the XML. With XML Spy, the XSL was created successfully and it was confirmed that the format of the XML was correct and efficient for the course application. XQuery was also used to understand XPath better. The next step was to create the application.

It was soon realized that two different views needed to be created. One view was to allow the user to see all the pre-requisites of the chosen course. The second view would show all the courses that can be taken. This view would be dependent on the user's given set of courses already taken. Due to the efficient format of the XML file, the same XML file can be used for both of these views. After deciding on the views and each view's function, the next step was designing the UI.

The UI would be created in a way that all the courses would not be visible at once. This was so the user does not become overwhelmed. For the first UI, where the pre-requisites of the selected course are seen, the use of three drop down menus was decided upon. Each set of courses had its own drop down menu. This also made it easy for the user to understand which set he or she will be choosing his or her course from. A CSS was created to give the drop down menus a unique look. For the second view, a single drop down menu was chosen where multiple courses could be chosen from. The length of this drop down was increased so more courses can be visible at once. For the first view, only one course from each set can be seen unlike the second view. The second view used a style sheet for the button the user clicks on once he or she is done selecting the courses he or she has taken already. The following figures will show some examples of how the views look and the results displayed starting with the first view's homepage in figure 10.


SJSU CS SJSU CS Courses Taken

CS Course Prerequisites

Please choose courses below and see their respective prerequisite course(s).

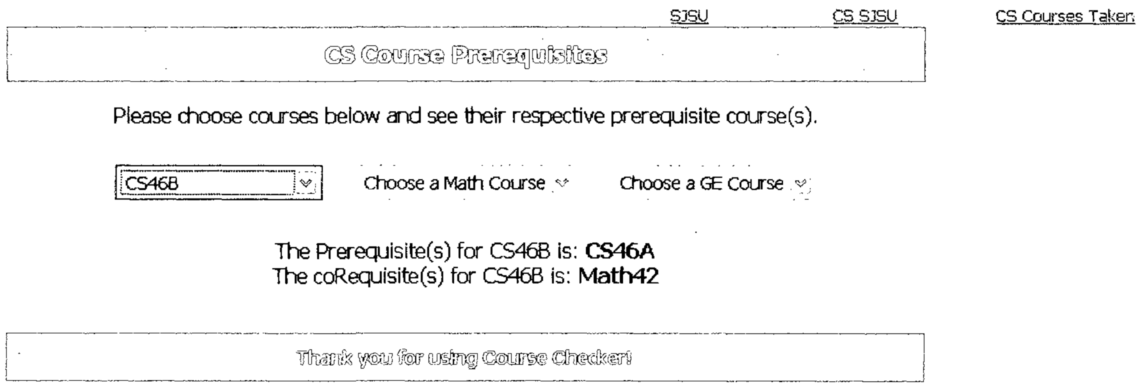
Choose a CS Course Choose a Math Course Choose a GE Course

Course prereq(s) will be listed here.

Thank you for using Course Checker!

Figure 10. Homepage of Course Pre-requisites

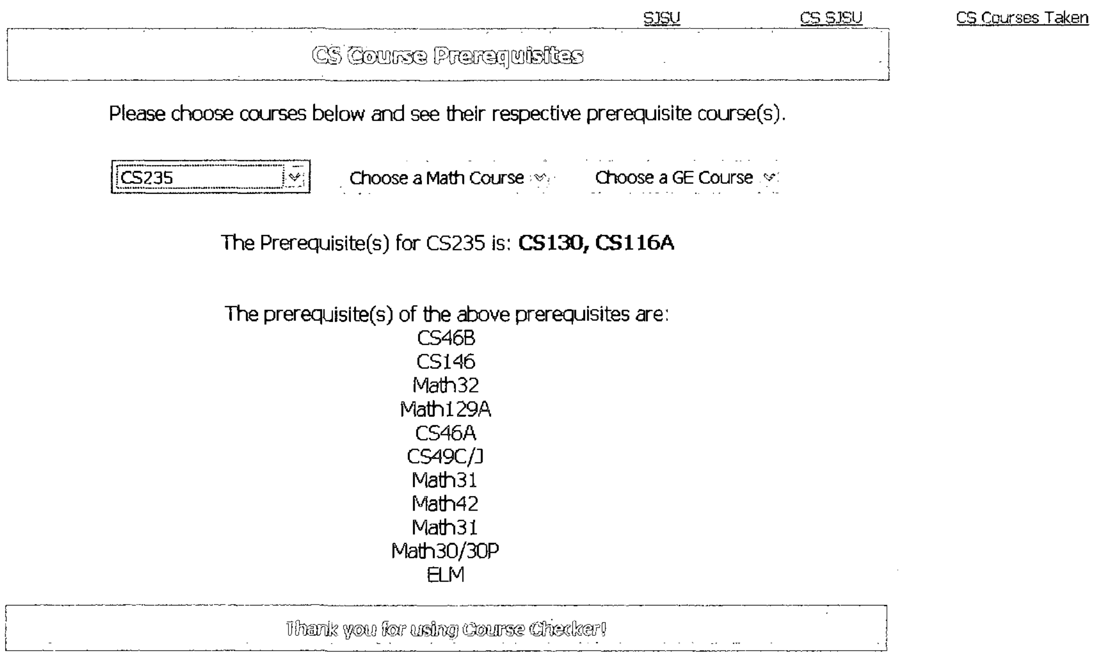
The figure above is the homepage to determining what the pre-requisites to a particular CS course is. As discussed above, there are three sets of drop down menus the user can make a selection from. Each set is in respect to the CS course catalog. Figure 11 shows the results if the user selects CS46B. Since the course also has co-requisites, those are displayed in the results. Figure 12 below shows the results to a graduate course CS235. It shows not only the direct pre-requisites of that course, but also all the cumulative pre-requisites. For example, the pre-requisites of the pre-requite course can also be seen. The color is different to set apart the distinction between a cumulative or direct pre-requisite.



Please choose courses below and see their respective prerequisite course(s).

The Prerequisite(s) for CS46B is: **CS46A**
The coRequisite(s) for CS46B is: **Math42**

Figure 11. Pre-requisite for the CS46B Course



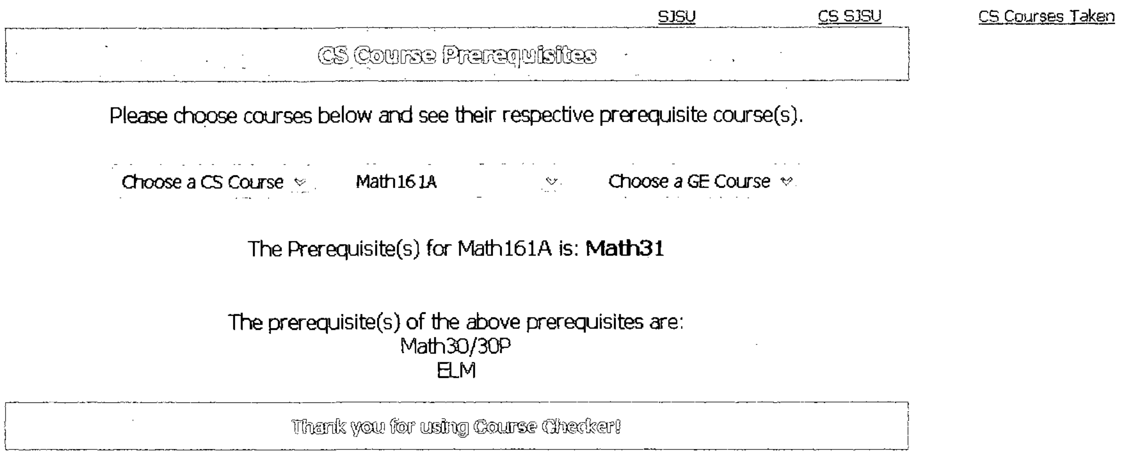
Please choose courses below and see their respective prerequisite course(s).

The Prerequisite(s) for CS235 is: **CS130, CS116A**

The prerequisite(s) of the above prerequisites are:
CS46B
CS146
Math32
Math129A
CS46A
CS49C/J
Math31
Math42
Math31
Math30/30P
ELM

Figure 12. Pre-requisite for the CS235 Course

The following two figures show the results when a course is chosen from the math and GE set. Figure 13 shows the results when the user selects the Math161A course. Figure 14 shows the results when the user selects the Phys51/71 course. The results are displayed in the same manner to create uniformity. Another point to see on the UI is the links in the upper right corner. The first link directs the user to the university's main website. The second link directs the user to San José State University's Computer Science website. The last link allows the user to traverse through the two different views. This minimizes the amount of clicks the user will have to do if he wants to go to any one of these three sites.


SJSU CS SJSU CS Courses Taken

CS Course Prerequisites

Please choose courses below and see their respective prerequisite course(s).

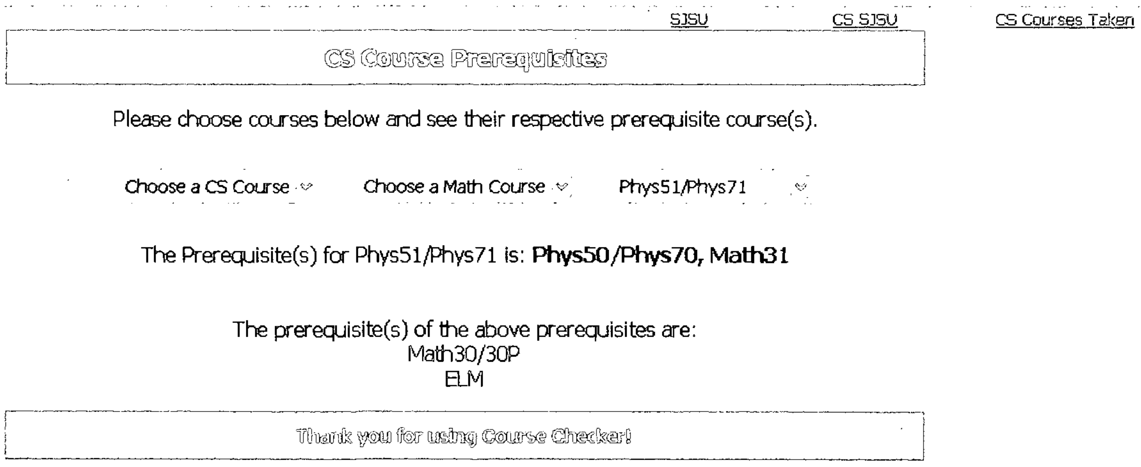
Choose a CS Course Math161A Choose a GE Course

The Prerequisite(s) for Math161A is: **Math31**

The prerequisite(s) of the above prerequisites are:
Math30/30P
ELM

Thank you for using Course Checker!

Figure 13. Pre-requisite of the Math161A Course


SJSU CS SJSU CS Courses Taken

CS Course Prerequisites

Please choose courses below and see their respective prerequisite course(s).

Choose a CS Course Choose a Math Course Phys51/Phys71

The Prerequisite(s) for Phys51/Phys71 is: **Phys50/Phys70, Math31**

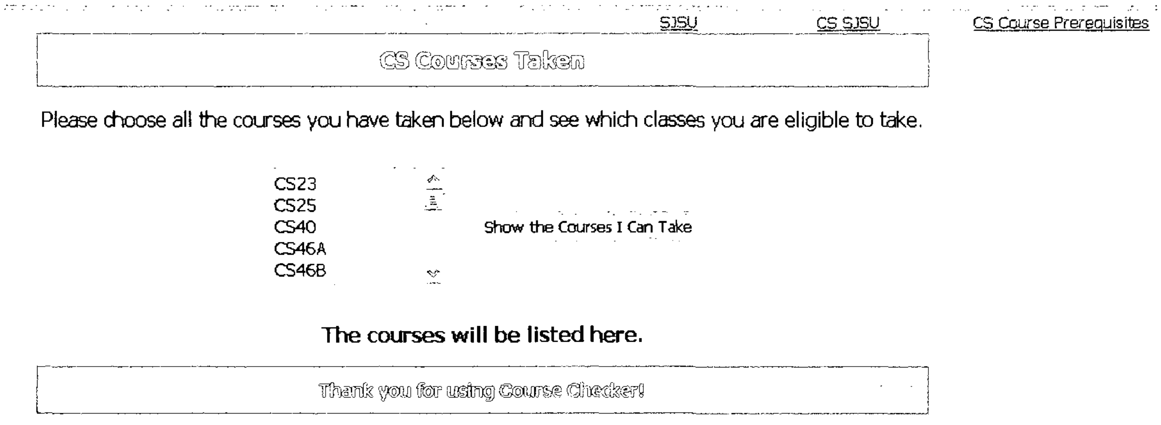
The prerequisite(s) of the above prerequisites are:
Math30/30P
ELM

Thank you for using Course Checker!

Figure 14. Pre-requisite of the Phys51/Phys71 Course

The second view is also similar to the first view in terms of colors and style used. It also contains the header and footer in the center of the page and the links in the upper right hand corner. This is again to create a sense of unity with the two views that the application offers. The figures shown below are snapshots of this view.

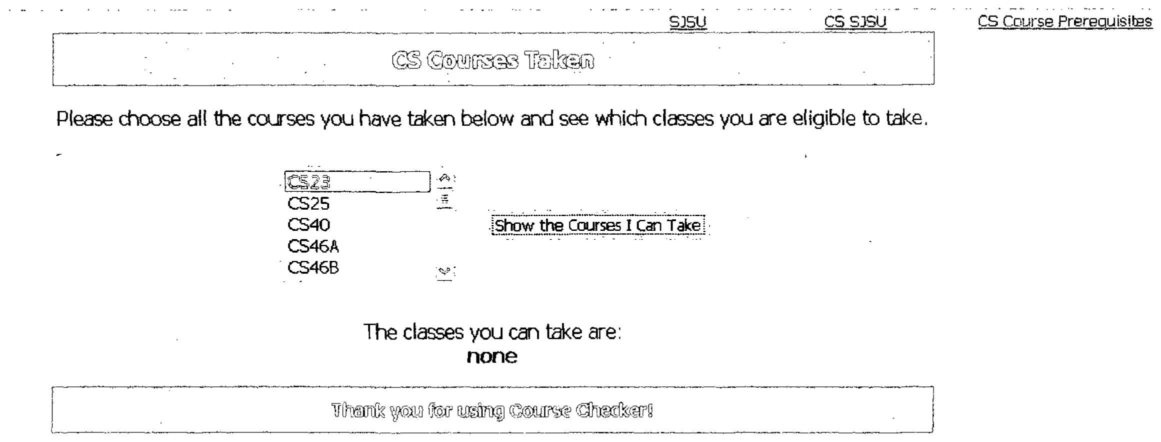
Figure 15 shows the homepage of courses taken. On the left side is a drop down menu where multiple courses can be selected by pressing the control key on the keyboard. The scroll bar allows the user to see all the courses available. Once the courses have been selected, the button on the right is clicked and the results are shown beneath the menu. The first figure below the homepage, figure 16, shows what is seen when the very first course is chosen and there are not any courses that can be taken. The following figure, figure 17, shows the results when one of the first courses in the CS undergraduate curriculum is chosen.



Please choose all the courses you have taken below and see which classes you are eligible to take.

The courses will be listed here.

Figure 15. Homepage of Courses Taken


SJSU CS SJSU CS Course Prerequisites

CS Courses Taken

Please choose all the courses you have taken below and see which classes you are eligible to take.

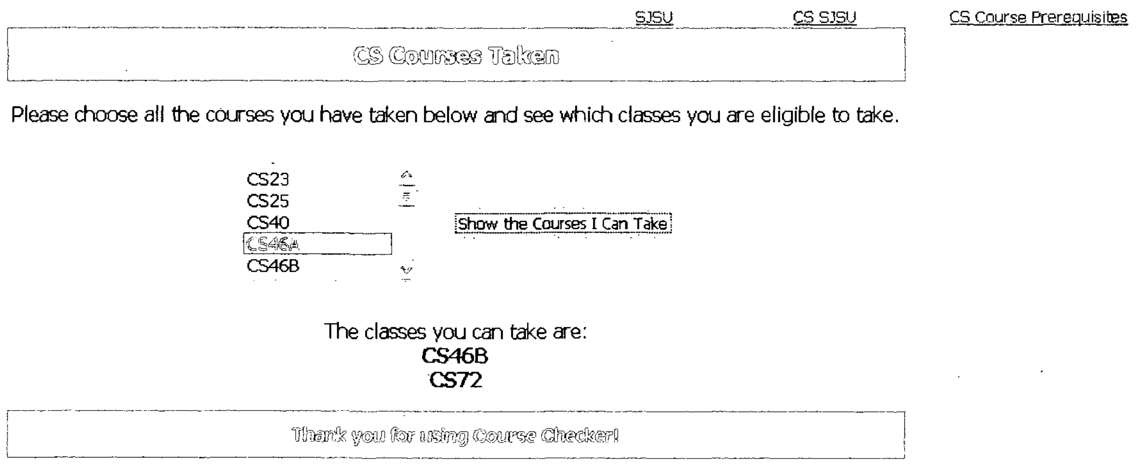
CS23
CS25
CS40
CS46A
CS46B

Show the Courses I Can Take

The classes you can take are:
none

Thank you for using Course Checker!

Figure 16. First Course Chosen


SJSU CS SJSU CS Course Prerequisites

CS Courses Taken

Please choose all the courses you have taken below and see which classes you are eligible to take.

CS23
CS25
CS40
CS46A
CS46B

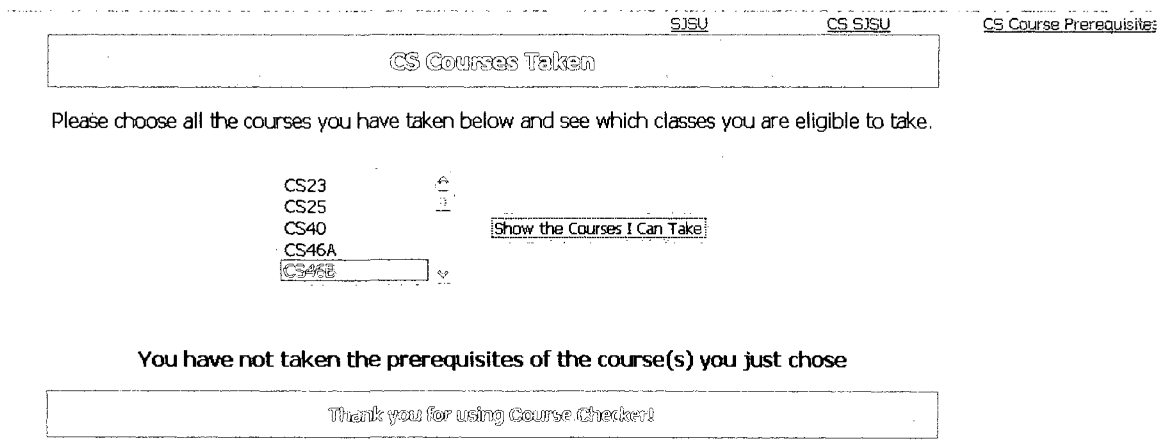
Show the Courses I Can Take

The classes you can take are:
CS46B
CS72

Thank you for using Course Checker!

Figure 17. One of the Initial Courses is Chosen

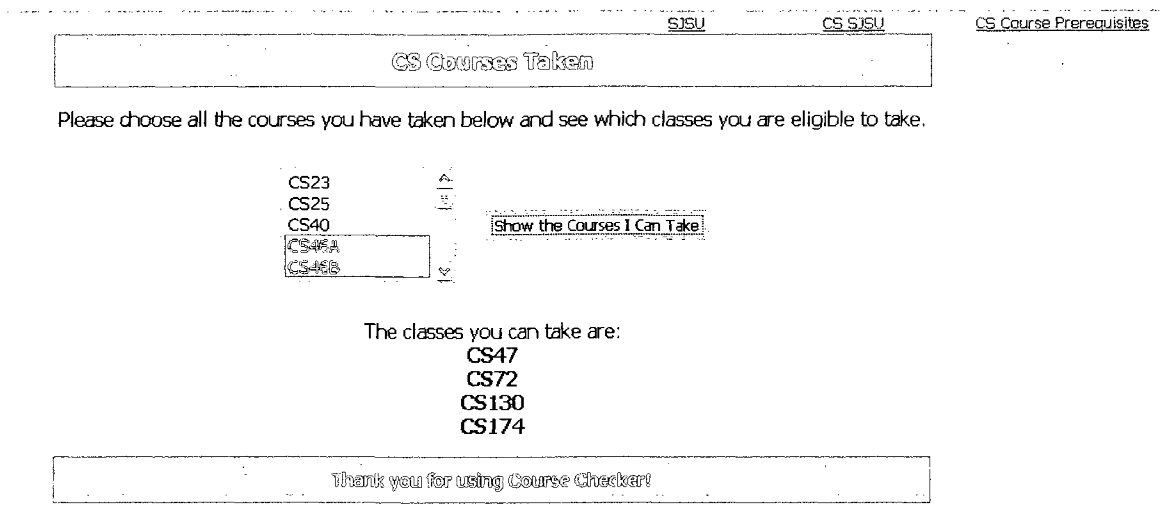
The figure, figure 18, below shows what happens when the user opts for a course in which he or she has not selected all the pre-requisite course(s). The system checks to make sure that the pre-requisites of the selected course(s) **have** also been met. If they have not been met, the user will see a message indicating so. For example, in figure 18, the only selected course is CS46B. Because the user must have also chosen CS46A, the error is shown. The following image, figure 19, shows the result when courses are selected correctly. Here, the pre-requisite CS46A is chosen along with CS46B therefore resulting in an answer. The result contains all the classes that can be taken by the user. Figure 20 shows what happens when the user just clicks on the button without selecting any course. The user will see the message as seen in the screenshot.



Please choose all the courses you have taken below and see which classes you are eligible to take.

You have not taken the prerequisites of the course(s) you just chose

Figure 18. Not All Pre-requisites Have Been Selected


SJSU CS SJSU CS Course Prerequisites

CS Courses Taken

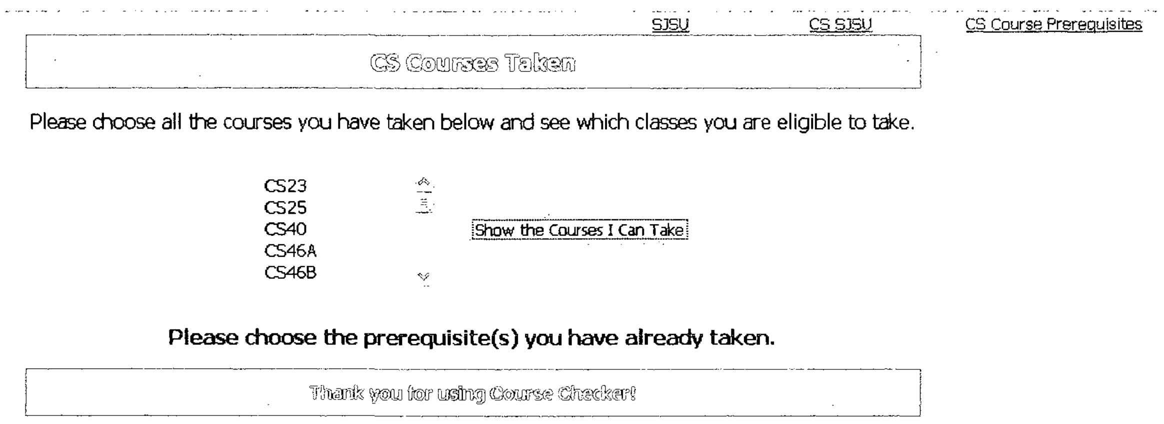
Please choose all the courses you have taken below and see which classes you are eligible to take.

CS23
CS25
CS40
CS46A
CS46B

The classes you can take are:
CS47
CS72
CS130
CS174

Thank you for using Course Checker!

Figure 19. Correct Selection of Courses


SJSU CS SJSU CS Course Prerequisites

CS Courses Taken

Please choose all the courses you have taken below and see which classes you are eligible to take.

CS23
CS25
CS40
CS46A
CS46B

Please choose the prerequisite(s) you have already taken.

Thank you for using Course Checker!

Figure 20. Button is Clicked Before Selecting Any Course

The courses application involves a simple CSP. The reason for this is there is only one type of constraint, the pre-requisites. For each view, this is the variable used. When a course is selected, the PHP script goes through the XML taking the pre-requisite into consideration. For example, for the first view, the course selected is searched in the XML file and the pre-requisites of that course are retrieved and displayed. The configurator goes one step more and keeps retrieving the pre-requisites of the pre-requisite courses until there are no more pre-requisites left. The second view uses all the selected courses and goes through each course's pre-requisites to see if the selected courses are in the set. If they are, then that course is added to the result set. Once finished, the output is displayed for the user. The next sub-section explains the complex application created.

Restaurant Reviews

Once the first application was completed, it was necessary to create a more complex system. Much research was done to finalize on a particular domain that the application should focus on. An area that is yet to be exploited is review. Currently, the best review site is Yelp. It has reviews in restaurants, shopping, food, beauty and spas, automotive, hotels, and many more. The top category is restaurants, so it was decided to work with a domain that has the most data. The UI of that category was looked at and many problems were seen right from the first view.

When the restaurants link is clicked, immediately the user sees many things at once that must be deciphered. The top five restaurants are at the top of the page. Underneath that is the ability for the user to refine the search by selecting a cuisine. Below that section are options to refine by other constraints. At the very bottom, there is a section about related searches. All the while on the right side of the page there is a column that is also divided into sections with more information. Although all that information is good, the user does become overwhelmed with too much information all at once. For these reasons, it was decided to create a restaurant review website that used constraint configuration as the backend on options that the user can use to refine the restaurant results. In addition, the Yelp website does not use the technology discussed in this paper of combining AJAX with existing technologies to quicken the results and better user experience.

There were many steps to build this application. Once the domain was finalized, it was necessary to build a prototype. This prototype served as a good reference point to the XML that would get created. The prototype showed the type of options that would be needed. Many restaurant and other types of websites and papers were researched to narrow down on the options to use. Once the prototype was agreed upon, the next step took the longest to complete.

Creating the XML required much time and thinking as it is the most crucial part of the system. The first step is how the restaurant data for each city was going to be obtained. The existence of a Yelp API came to notice and this solved some of the problems. It allowed any programmer to access the top twenty restaurants of each city. No more than twenty businesses could be accessed. This did pose a problem as far as the amount of data obtained. This issue was overcome by obtaining all the cities that are available on Yelp. The cities were obtained by using PHP and accessing the HTML contents of each state. The use of regular expressions allowed the correct data, only the cities, to be obtained from the HTML pages. To address the memory issue that arises with loading such large XML files, it was decided to create a separate XML file for each state. This was reasonable as only one city's restaurants are needed at a time. The next topic to be addressed was the XML schema.

The XML schema took many revisions. The reason for this was every time a schema was modified, issues such as how the PHP script would traverse through the XML would reject it. It was agreed upon that the data obtained from the API would be stored as attributes to each restaurant element. Each restaurant would be in a particular city. This seemed to be satisfactory.

However, when the API results were looked at, it was noticed that many items were missing. The API provided only the basic attributes needed but most of the option ones were not included. From the API, the attributes extracted from each business result are seen in table 4.

Table 4 API Attributes Extracted

Attribute Name	**Purpose**
name	The name of the restaurant.
Address	The address of the restaurant.
city	The city the restaurant is in.
State	The state the restaurant is in.
Zip	The zipcode of the restaurant.
Latitude	The latitude of the restaurant.
Longitude	The longitude of the restaurant.
Neighborhoods	The neighborhoods of the city.
phone	The restaurant's phone number.
avg_rating	Average rating for restaurant.
Review_count	Number of reviews for restaurant.
Rating_img_url_small	Rating image for restaurant.
url	Link to the restaurant page in yelp.
photo_url	Link to the restaurant's photo to display.

The table above has the names of the information extracted from the API.

The left column indicates the name of the attribute from the API. The right column explains what the attribute is for in respect to a restaurant. The API mostly returned location information about a restaurant. As can be seen from table 4, none of the restaurant constraints like the meals it serves and its price range are in the results. These are crucial pieces of information that needed to be obtained. Since the URL of the restaurant was obtained from the API, the script then acquired the HTML contents of that restaurant web page using a regular expression. The expression extracted the option information. Table 5 shows the type of options extracted from each restaurant's web page.

Table 5 Restaurant Option Information

Option Name	**Value(s)**
price	$/$$/$$$/ $$$$
AcceptsCreditCards	Yes/No
Parking	Street/Garage/Valet/Private Lot/Validated
Attire	Formal/Casual
GoodForGroups	Yes/No
GoodForKids	Yes/No
TakesReservations	Yes/No
Delivery	Yes/No
WaiterService	Yes/No
WheelchairAccessibility	Yes/No
OutdoorSeating	Yes/No
MealType	Breakfast/Brunch/Lunch/Dinner/ LateNightSnack
Alcohol	Full Bar/Beer & Wine Only/Happy Hour

These options were also added as attributes to a restaurant element. The above table shows the value of the attributes that are added. The left column is the name of the attribute and the right column is the range of the attribute's value. It was after the HTML parsing, that the information extracted for each restaurant was complete. A snippet of the XML schema can be scene below in figure 21. The root element for the schema is *cityRestaurants*. It contains *city* elements. Each city contains *restaurant* elements. The attributes of the city and restaurant elements can be seen in the figure below. Because there are many restaurant attributes, they have been displayed in two columns.

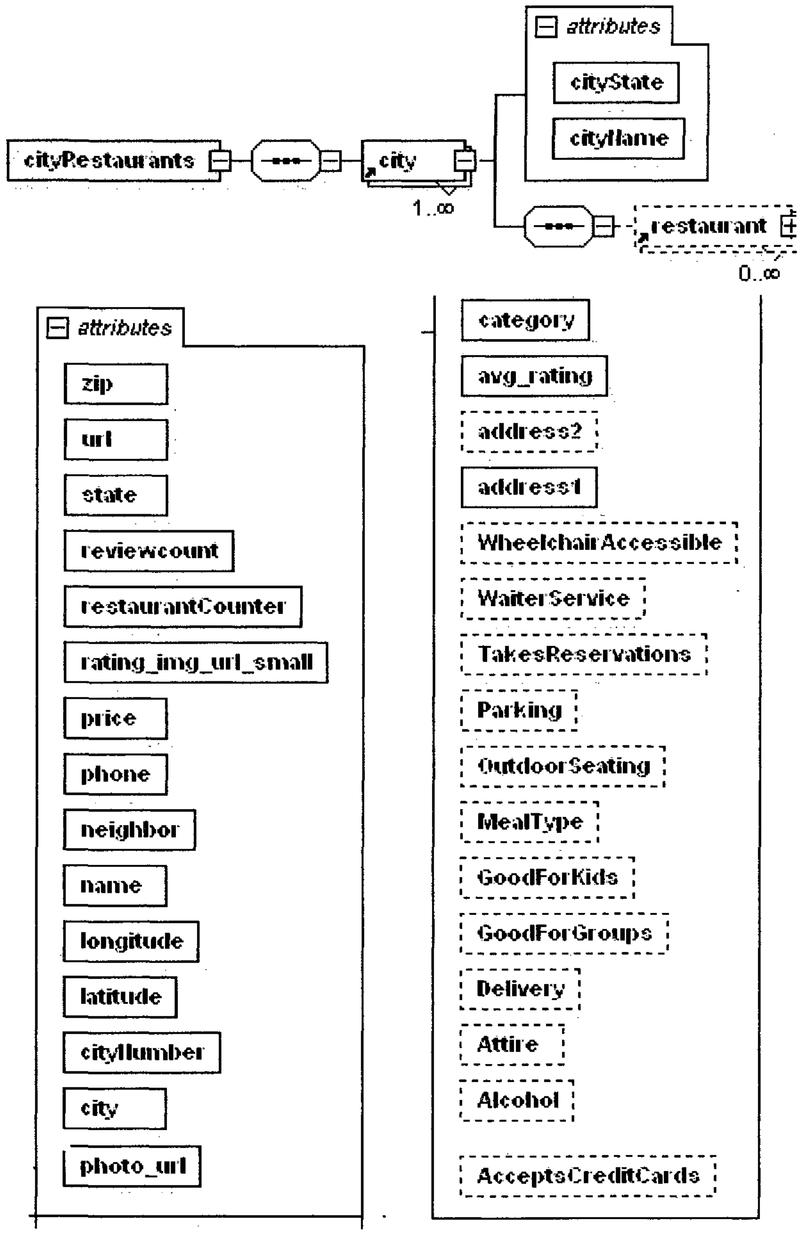
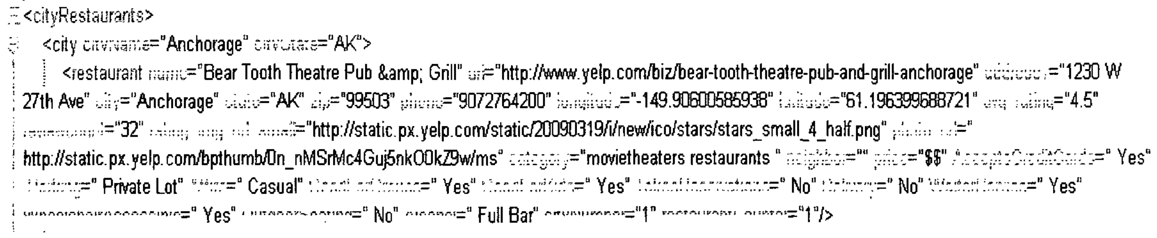


Figure 21. Restaurant XML Schema

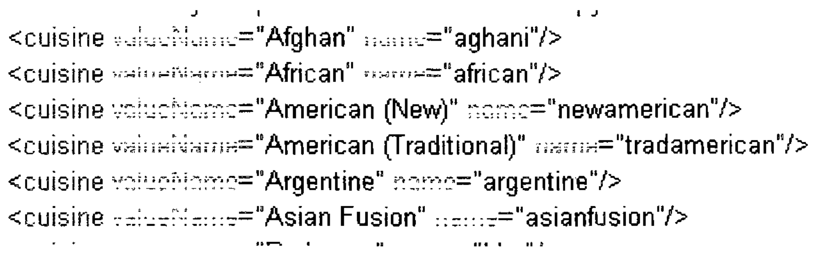
Figure 28 shows how the schema is going to be able to map the constraints. The mapping is not difficult; therefore, the server-side scripting won't be either. As aforementioned, each restaurant is contained in a city element which is in a state XML file. When the user enters a location, the corresponding state XML file is opened and parsed accordingly. Figure 22 below shows a snippet of what a corresponding XML file would look like. Each restaurant has all the attributes shown above in the schema figure. The example here is a restaurant in Anchorage, AK. All the XML files of each state follow this format.



```
<cityRestaurants>
    <city cityname="Anchorage" citystate="AK">
        <restaurant name="Bear Tooth Theatre Pub &amp; Grill" url="http://www.yelp.com/biz/bear-tooth-theatre-pub-and-grill-anchorage" address="1230 W
27th Ave" city="Anchorage" state="AK" zip="99503" phone="9072764200" longitude="-149.90600585938" latitude="61.196399688721" avg_rating="4.5"
[illegible]="32" [illegible]="http://static.px.yelp.com/static/20090319/i/new/ico/stars/stars_small_4_half.png" [illegible]="
http://static.px.yelp.com/bpthumb/0n_nMSrMc4Guj5nkO0kZ9w/ms" category="movietheaters restaurants " neighbor="" price="$$" [illegible]=" Yes"
[illegible]=" Private Lot" [illegible]=" Casual" [illegible]=" Yes" [illegible]=" Yes" [illegible]=" No" [illegible]=" No" [illegible]=" Yes"
[illegible]=" Yes" [illegible]=" No" [illegible]=" Full Bar" [illegible]="1" [illegible]="1"/>
```

Figure 22. Restaurant XML File

Another XML file was made for the display of the cuisines available. This file was traversed through for the cuisine option. A snippet of the cuisine XML file is shown below in figure 23.



```
<cuisine valueName="Afghan" name="aghani"/>
<cuisine valueName="African" name="african"/>
<cuisine valueName="American (New)" name="newamerican"/>
<cuisine valueName="American (Traditional)" name="tradamerican"/>
<cuisine valueName="Argentine" name="argentine"/>
<cuisine valueName="Asian Fusion" name="asianfusion"/>
```

Figure 23. Cuisine XML Snippet.

The XML snippet shown above contains cuisine elements. The elements contain two attributes, the *valueName* and *name*. The *valueName* attribute is the name that is seen by the user. The *name* attribute is the value of the constraint used in the constraint configurator. The last XML that needed to be created was the ingredient mapping file.

Because the Main Ingredient option was a new feature added, an ingredient attribute was not added to each restaurant element. Instead, a constraint mapping was created between ingredient and cuisine. This created a unique constraint problem. As stated by Wikipedia, "constraint satisfaction is the process of finding a solution to a set of constraints that impose conditions that the variables must satisfy" (Constraint Satisfaction, par. 1).

The problem to be solved for this constraint is the many-to-many mapping that is resulted. A cuisine contains multiple ingredients and an ingredient belongs to multiple cuisines. Take the ingredient rice for example. It belongs in multiple cuisines such as Asian fusion, Mexican, and Greek. Figure 24 below shows the relationship of many-to-many mapping.

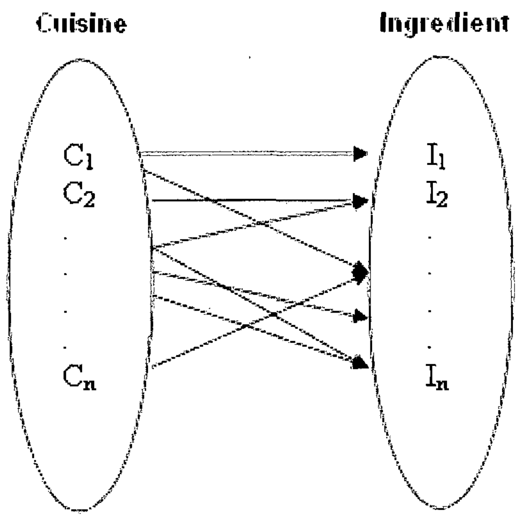


Figure 24. Cuisine-Ingredient Mapping

The figure above shows more clearly the mapping between cuisine and ingredient. XML is known to provide one-to-one and one-to-many mapping, but some tricks are required to represent many-to-many mapping. The reason for this is an efficient format to represent the many-to-many relationship needs to be presented to reduce the searching time the constraint configurator will take on the XML. Bill Lewis, a Data Architect at IBM, wrote in a newsletter a way to represent many-to-many mapping in XML. He used the many-to-many mapping seen in a party-location relationship. Like the cuisine-ingredient mapping shown above, a party could have many locations and many locations are associated with different parties. To accommodate for this relationship, he introduced a dependent associative entity. This entity contains a party and location element (Lewis). As a result, the following schema seen in figure 25 is written.

```
<xs:element name="Association">
   <xs:complexType>
         <xs:sequence maxOccurs="unbounded">
            <xs:element ref="Cuisine"/>
            <xs:element ref="Ingredient"/>
            <xs:element ref="Cuisine_Ingredient_Association"/>
         </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="Cuisine">
   <xs:element name="cuisineID" type="xs:string"/>
   <xs:element name="cuisineName" type="xs:string"/>
</xs:element>
<xs:element name="Ingredient">
   <xs:element name="ingredientID" type="xs:string"/>
   <xs:element name="ingredientName" type="xs:string"/>
</xs:element>
<xs:element name="Cuisine_Ingredient_Association">
   <xs:element name="cuisineID" type="xs:string"/>
   <xs:element name="ingredientID" type="xs:string"/>
</xs:element>
```

Figure 25. Cuisine-Ingredient XML Schema

The schema seen above shows the three main elements, *Cuisine, Ingredient,* and *Cuisine_Ingredient_Association.* The cuisine elements are all unique, along with the ingredient elements; there are no duplicates. The *Cuisine_Ingredient_Association* also has unique sets of values. After the ingredient and cuisine elements had been written, the associations between them had to be made. Wikipedia was referenced as the place to find the main ingredients for each cuisine. For example, if the ingredient rice is chosen, the following figure,

figure 26, shows the flow of the constraint configurator.

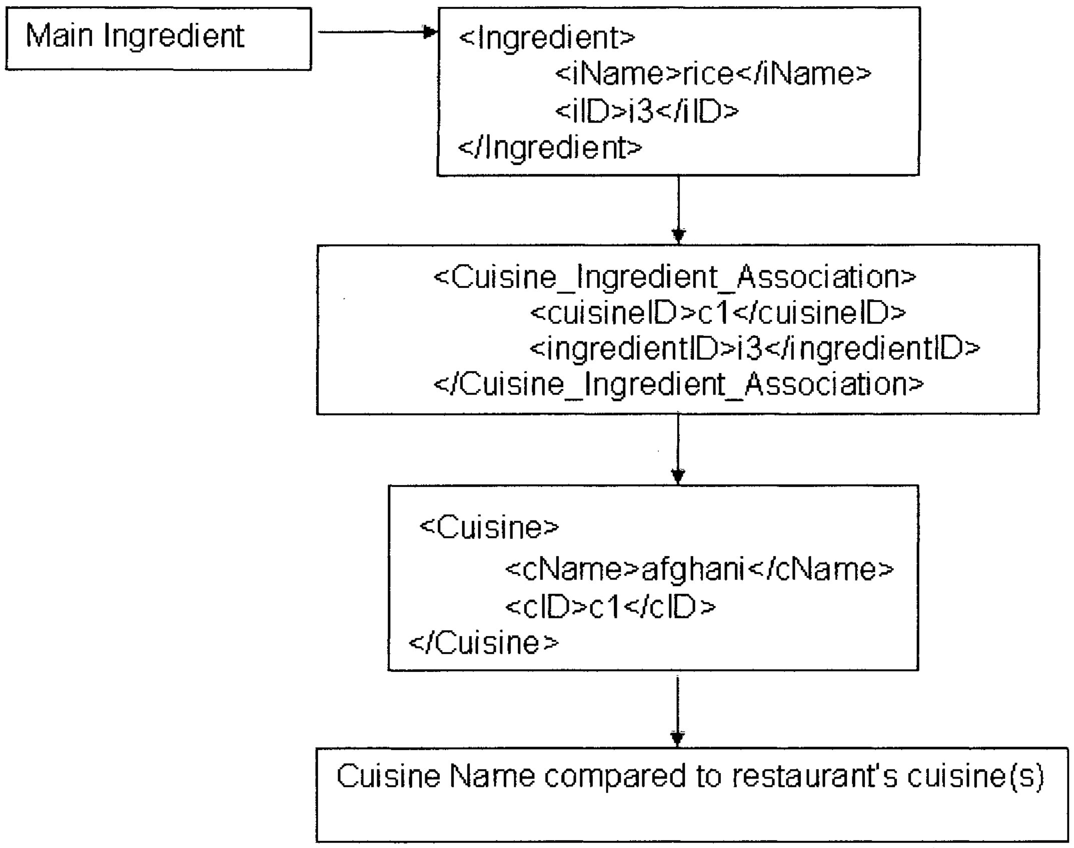

Main Ingredient
<Ingredient>
<iName>rice</iName>
<iID>i3</iID>
</Ingredient>
<Cuisine_Ingredient_Association>
<cuisineID>c1</cuisineID>
<ingredientID>i3</ingredientID>
</Cuisine_Ingredient_Association>
<Cuisine>
<cName>afghani</cName>
<cID>c1</cID>
</Cuisine>
Cuisine Name compared to restaurant's cuisine(s)

Figure 26. Ingredient Mapping Flow

When the user selects an ingredient, like rice, the constraint configurator goes to the Association XML file and using XPath finds the id value of that ingredient. It then attains all the *Cuisine_Ingredient_Association* elements with that id. The above figure shows one of the *Cusine_Ingredient_Association* elements found by the configurator. The next step is that the corresponding cuisine names are found. It is checked if the restaurant has that cuisine in its attribute. If the cuisine is contained in the restaurant, then it is added to the results. Otherwise, the configurator moves on to the next *Cuisine_Ingredient_Association* element. In this way, only the restaurants with that cuisine are shown to the user.

The question of how efficient this XML is may arise to one's mind. First of all, the *Cuisine_Ingredient_Association* elements were made using a form. This was the most efficient and quick way of creating the elements. Each cuisine was looked up in Wikipedia and the elements were made accordingly. Figure 27 shows a sample screenshot of the form.

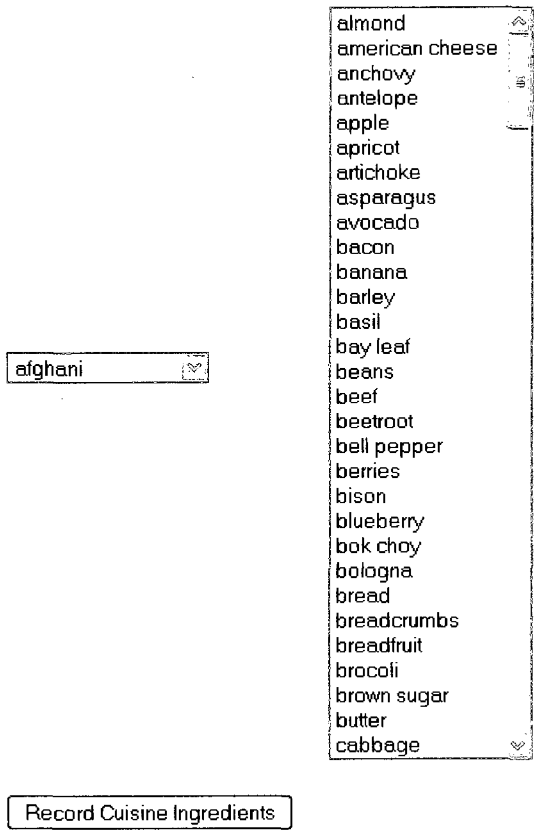


Figure 27. Creating *Cuisine_Ingredient_Association* Elements

For each cuisine, the set of ingredients were chosen. Once done, the Record Cuisine Ingredients button was clicked. This created the XML elements with the content added to the Association XML document. At the end, there was one Association XML document that contained all the *Cuisine_Ingredient_Association* elements along with the *Cuisine* and *Ingredient* elements.

The question about the size of the XML document may be a concern.

However, with the XML document design, this problem is overcome quite easily. At worst case there can be 82 X 215 *Cuisine_Ingredient_Association* elements in the XML. There are 82 cuisines and 215 Ingredients. This will result to 17, 630 elements. If XPath was not used, traversing through these elements would have been cumbersome. However, at worst case, the configurator will have to test 82 *Cuisine_Ingredient_Association* elements in one search cycle. The reason for this is that an ingredient cannot be duplicated in a cuisine, so at worst case an ingredient is in every cuisine. XPath allows the ability to only look at elements where the condition has been met. By looking at figure 26 it can be seen that the XPath condition is the ingredient Id when attaining *Cuisine_Ingredient_Association* the elements.

As mentioned earlier, one of the goals of creating this application was to make it user-friendly. This was done by presenting minimal and only necessary material to the user. The categorization of the menu items further decreased the web page space used at one time. With the side menu, the user was able to see the results being refined instantaneously. The colors chosen were to create calm and controlling emotions to the user. The use of color coordinated buttons further solidified the feelings. It was of key importance to make the users feel the application is geared towards them and this was achieved.

The constraint configuration for the options was crucial in creating pleasing results. This was dependent on the XML design. Depending on the option selected, a certain mechanism in the configurator was chosen. The figure below, figure 28, shows the option to XML mapping.

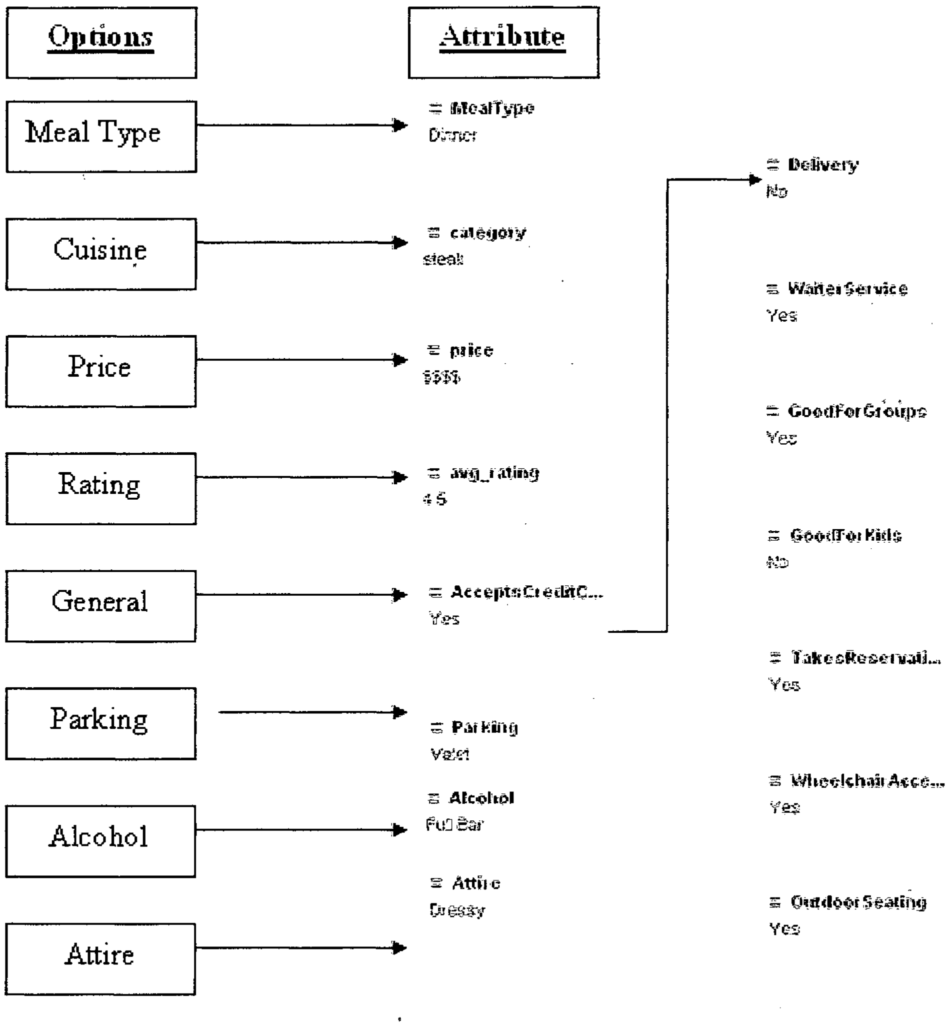
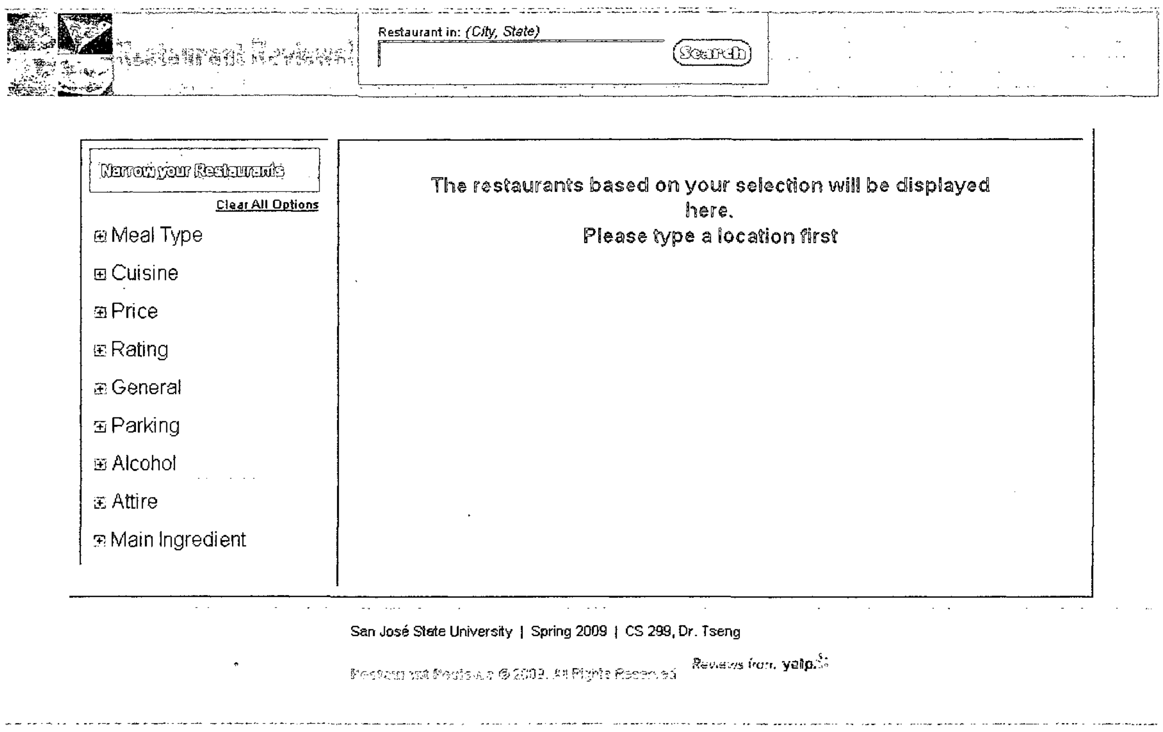


Figure 28. Options to Restaurant Attribute Mapping

When a constraint from one of the options is chosen, the constraint configurator goes to the respective restaurant attributes to perform the constraint algorithm. If the comparison is true, then that restaurant is added to the results. Otherwise it isn't. The restaurants from which the configurator searched from are only the ones from the location specified by the user. Again, XPath is used to get only the required data so that traversing through all the nodes is not required. The algorithm to the ingredient selection is a bit different than when the user selects the other options. This can be seen in figure 28.

To add more information to the web application, Google Maps is also used. The Google Maps API is used to plot the restaurant results. The map is above the result set with the corresponding numbering. Each of those number's info window contains the restaurant's name, address and phone number. The result set below the map also contains an image of the restaurant. Autocomplete is added in the main ingredient option. The autocomplete is in the text box where users can type in ingredient(s) that are not seen. Figure 34 below shows how the autocomplete looks like. The autocomplete code is taken from the Yahoo User Interface Library. The JavaScript files needed are included in the HTML file. The only additional information needed is an array of the ingredients.

Below are screen shots of the UI. The homepage, figure 29, is first shown followed by different cases.

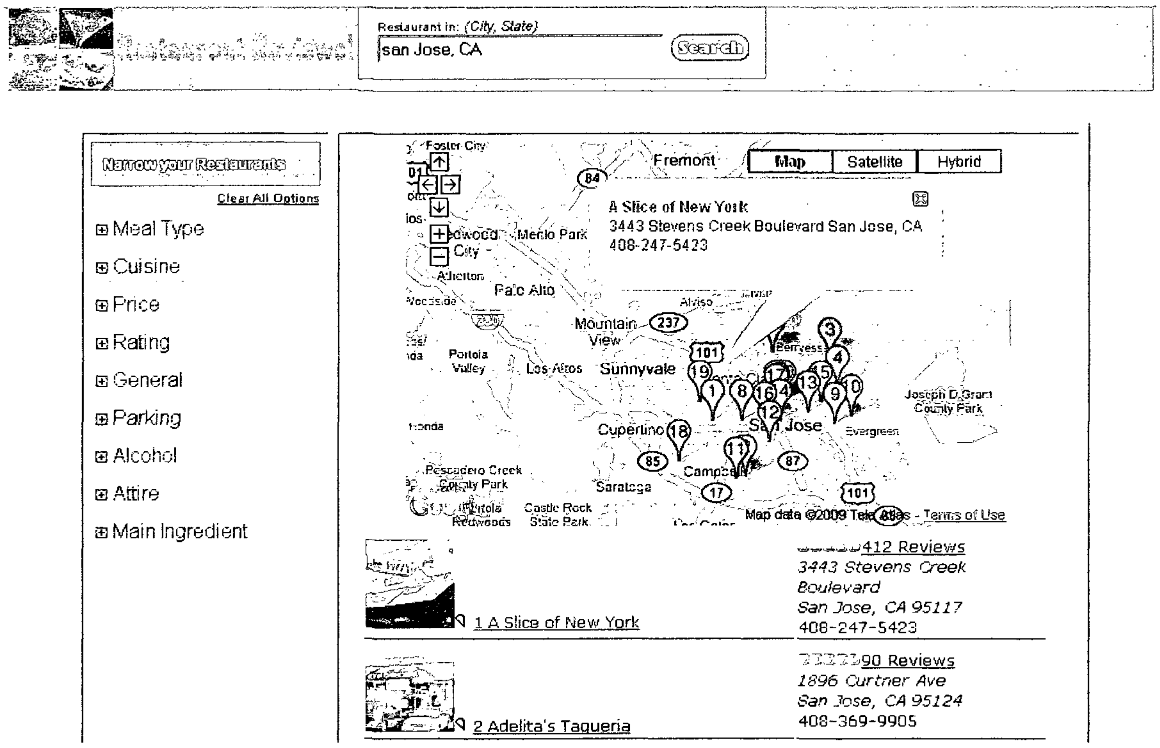


The restaurants based on your selection will be displayed here.
Please type a location first

San José State University | Spring 2009 | CS 299, Dr. Tseng

Figure 29. Restaurant Review Homepage

The application is designed to have a header and footer. The header contains the restaurants review image. It is also a link for the user to go back to the homepage. The user enters the location in the format: City, State. The state is in the US postal code format. The footer contains information of the site and has a link to the homepage of Yelp. The center of the application is divided into two parts. The left column is the options the user chooses from. To select a constraint from an option, the user clicks on the image with a plus sign. The option will expand and the constraints associated with it will be seen. The selection of a constraint will see immediate results on the right side. Figure 30 shows the web page when a location has been selected.



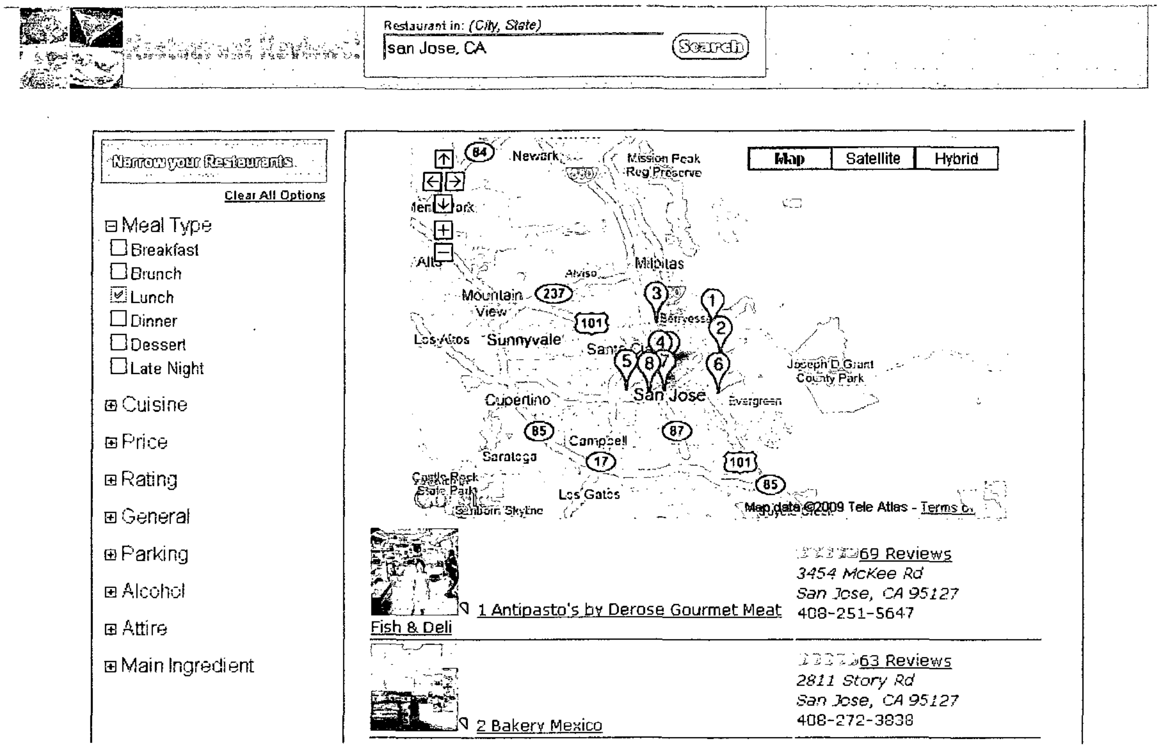
A Slice of New York
3443 Stevens Creek Boulevard San Jose, CA
408-247-5423

1 A Slice of New York
412 Reviews
3443 Stevens Creek Boulevard
San Jose, CA 95117
408-247-5423

2 Adelita's Taqueria
90 Reviews
1896 Curtner Ave
San Jose, CA 95124
408-369-9905

Figure 30. San Jose, CA Restaurants

The above figure shows the restaurants that are seen once the location is typed in the text field. The map is shown at the top with numbers on the markers. The markers are in reference to the restaurant information below the map. The information window for restaurant number one is also seen in the figure. The restaurants are in alphabetical order. The user sees the restaurant name and an image on the left side and on the right side the user sees more information. The stars it has received from the reviews along with the number of reviews are on the first line. After that, is the address of the restaurant. The last line contains the phone number of the restaurant. One thing to note is that the text field input is not case sensitive. What is needed between the city and state is a comma, but the capitalization does not matter. In other words, typing sAn jOse, ca or san jose, ca will result in the same set of restaurants. Figure 31 shows a screen shot of a refinement.


Restaurant in: (City, State)
san Jose, CA
Search

Narrow your Restaurants
Clear All Options
⊟ Meal Type
☐ Breakfast
☐ Brunch
☑ Lunch
☐ Dinner
☐ Dessert
☐ Late Night
⊞ Cuisine
⊞ Price
⊞ Rating
⊞ General
⊞ Parking
⊞ Alcohol
⊞ Attire
⊞ Main Ingredient

Map | Satellite | Hybrid

1 Antipasto's by Derose Gourmet Meat Fish & Deli
69 Reviews
3454 McKee Rd
San Jose, CA 95127
408-251-5647

2 Bakery Mexico
63 Reviews
2811 Story Rd
San Jose, CA 95127
408-272-3838

Figure 31. Refining by Meal Type

The above figure is a set of restaurants of San Jose, CA refined by the constraint Lunch in the Meal Type Option. If compared to figure 30, one can see that the results have changed. The first restaurant in both the figures is not the same. This can be seen easily in the map. It is seen that restaurants that serve Lunch are shown only versus all the restaurants in figure 30. The rest of the options offer the same type of refinement in the results. The General option's constraints are more of a yes or no value. For example, if the user selects Takes Reservations that means restaurants that takes reservations should be in the result set. The figure below, figure 32, shows a sample.

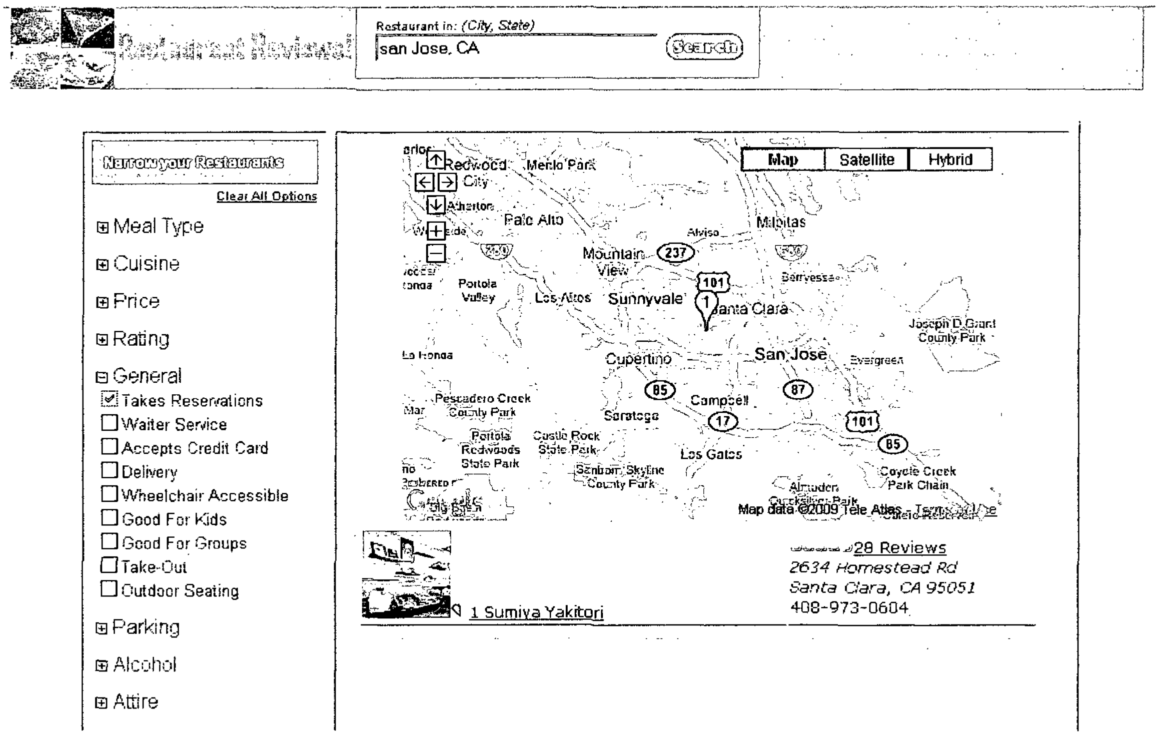
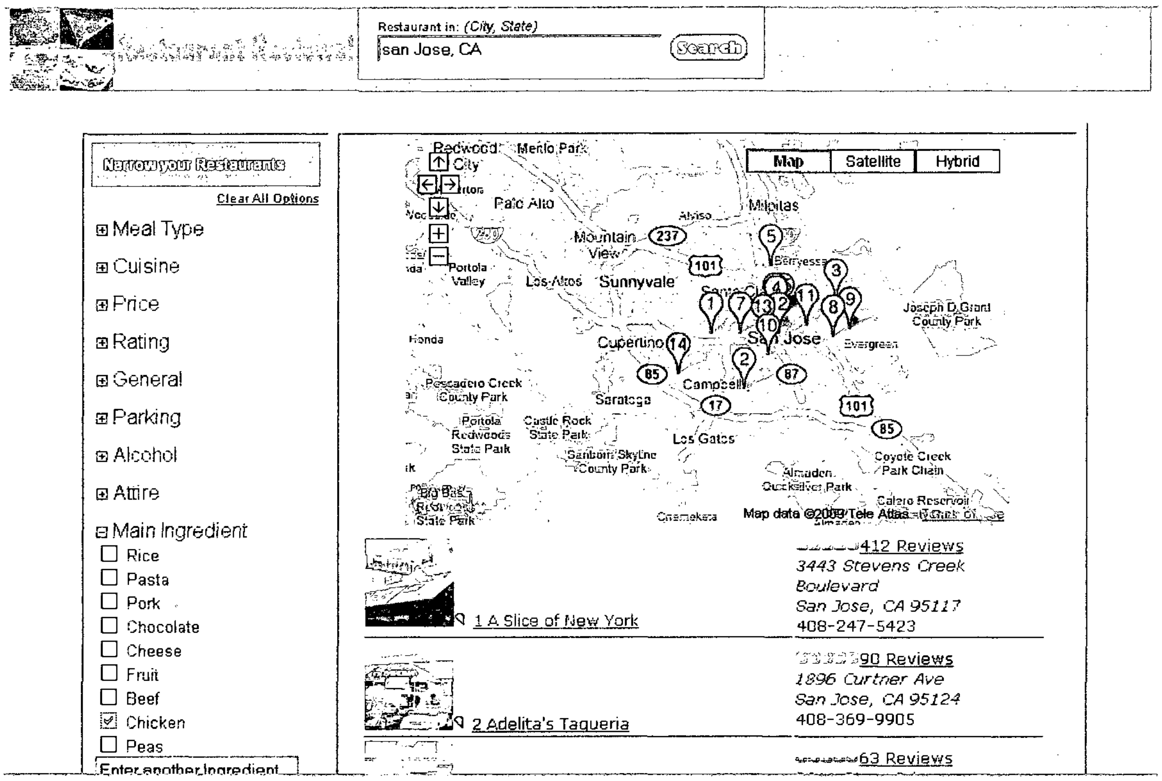


Figure 32. Refine by Takes Reservations

When the San Jose, CA restaurants are refined by the general constraint Takes Reservations, only one restaurant results, Sumiya Yakitori. The next figures, figure 33, 34, 35, show the Main Ingredient option constraints. These figures will show what is seen when a constraint being displayed is selected and when one constraint created by the user is also selected and used to refine the result set. Different set of events happen in the backend depending on which constraint is chosen in the Main Ingredient Option. The constraint is either selecting a pre-defined constraint, or creating a new one. Figure 33 shows the selection of a pre-defined constraint called Chicken.



Restaurant in: (City, State) san Jose, CA

- [ ] Rice
- [ ] Pasta
- [ ] Pork
- [ ] Chocolate
- [ ] Cheese
- [ ] Fruit
- [ ] Beef
- [x] Chicken
- [ ] Peas

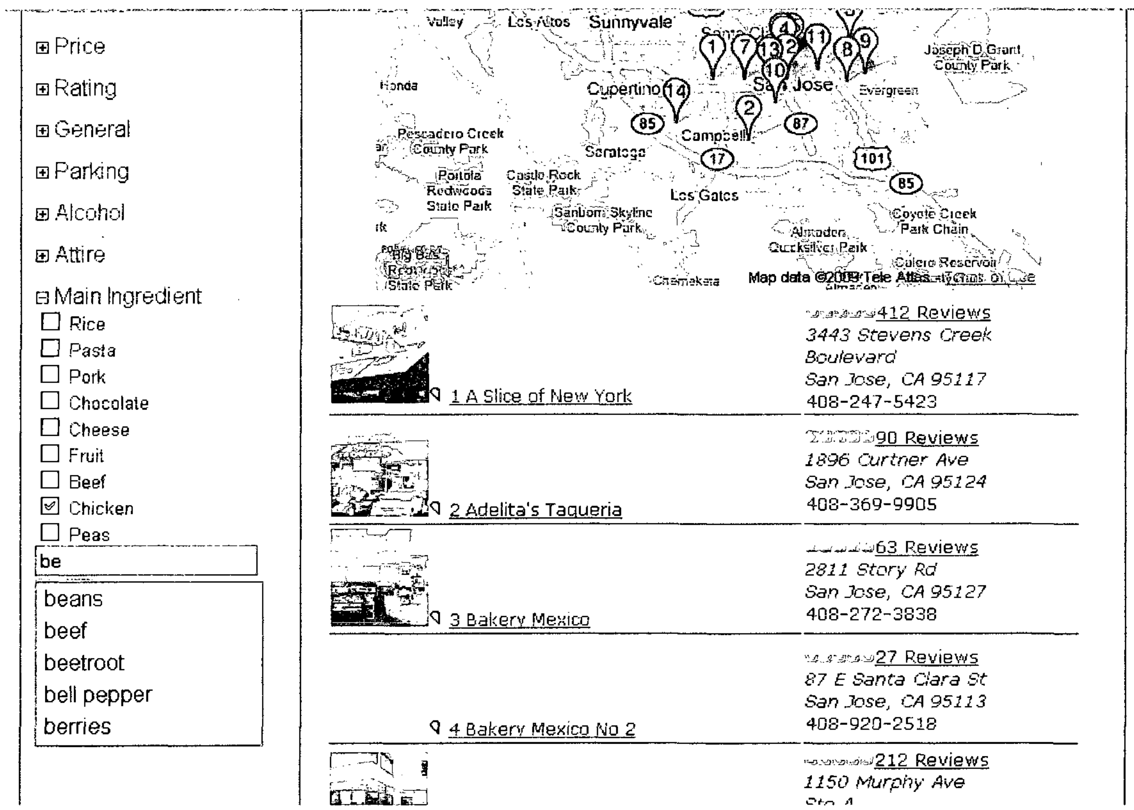
Enter another Ingredient

1 A Slice of New York — 412 Reviews, 3443 Stevens Creek Boulevard, San Jose, CA 95117, 408-247-5423

2 Adelita's Taqueria — 90 Reviews, 1896 Curtner Ave, San Jose, CA 95124, 408-369-9905

63 Reviews

Figure 33. Chicken Constraint in San Jose, CA



⊞ Price
⊞ Rating
⊞ General
⊞ Parking
⊞ Alcohol
⊞ Attire
⊟ Main Ingredient
☐ Rice
☐ Pasta
☐ Pork
☐ Chocolate
☐ Cheese
☐ Fruit
☐ Beef
☑ Chicken
☐ Peas

be

beans
beef
beetroot
bell pepper
berries

1 A Slice of New York — 412 Reviews, 3443 Stevens Creek Boulevard, San Jose, CA 95117, 408-247-5423

2 Adelita's Taqueria — 90 Reviews, 1896 Curtner Ave, San Jose, CA 95124, 408-369-9905

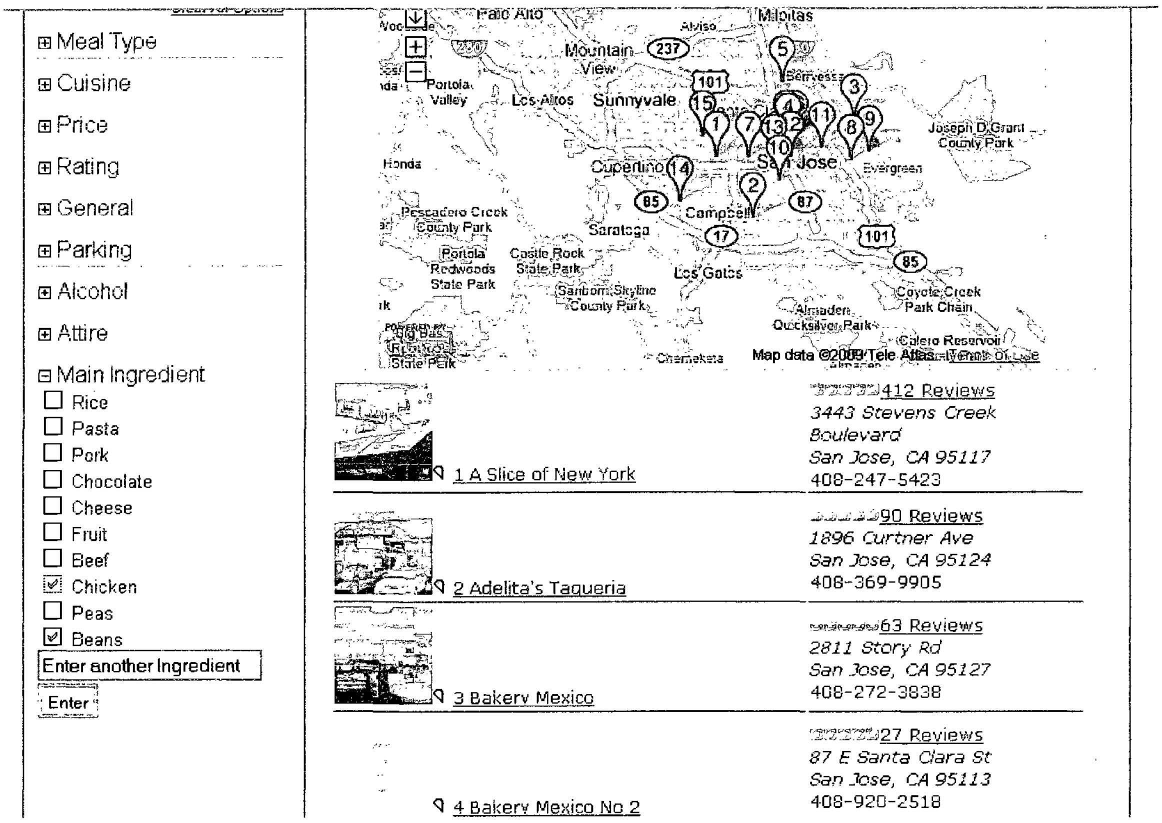
3 Bakery Mexico — 63 Reviews, 2811 Story Rd, San Jose, CA 95127, 408-272-3838

4 Bakery Mexico No 2 — 27 Reviews, 87 E Santa Clara St, San Jose, CA 95113, 408-920-2518

212 Reviews, 1150 Murphy Ave

Figure 34. Adding Beans to the Main Ingredient Constraint

The word beans is typed as another main ingredient constraint. Notice the autocomplete that is seen while the user types the constraint to be added. The user can see what constraints are available. This event triggers the configurator and then a display function to add the new ingredient to the visible Main Ingredient Option constraints as seen in figure 35.



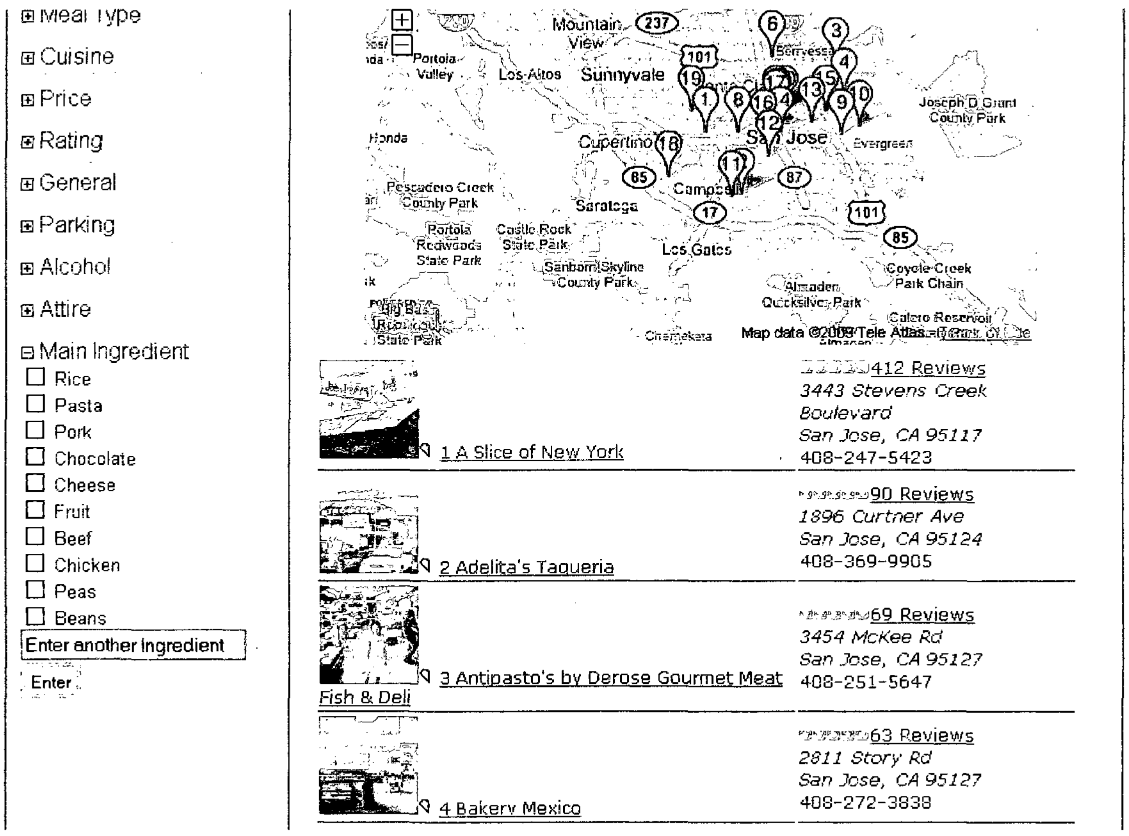
⊞ Meal Type
⊞ Cuisine
⊞ Price
⊞ Rating
⊞ General
⊞ Parking
⊞ Alcohol
⊞ Attire
⊟ Main Ingredient
☐ Rice
☐ Pasta
☐ Pork
☐ Chocolate
☐ Cheese
☐ Fruit
☐ Beef
☑ Chicken
☐ Peas
☑ Beans
Enter another Ingredient
Enter

1 A Slice of New York	412 Reviews 3443 Stevens Creek Boulevard San Jose, CA 95117 408-247-5423
2 Adelita's Taqueria	90 Reviews 1896 Curtner Ave San Jose, CA 95124 408-369-9905
3 Bakery Mexico	63 Reviews 2811 Story Rd San Jose, CA 95127 408-272-3838
4 Bakery Mexico No 2	27 Reviews 87 E Santa Clara St San Jose, CA 95113 408-920-2518

Figure 35. Beans Constraint Added to Main Ingredient

The figure above shows the end result of adding beans to the Main Ingredient option. The results have been refined and the beans constraint has been added. Notice in figure 35, that beans is already selected as it is a constraint chosen. Also, the display of the word is unique. To continue with the look, only the first letter of each word is a capital Letter. This is done by the ucwords and strtolower methods in PHP. The latter method is done first to put the entire string to all lowercase letter and then make the first letter in each word capital.

The beans value entered is still there when the clear all options link is clicked as seen in figure 36. It is not until the home link is clicked that the constraint is not seen anymore. This is because the user may want to select the ingredient again, so the constraint is still visible.



⊞ Meal Type
⊞ Cuisine
⊞ Price
⊞ Rating
⊞ General
⊞ Parking
⊞ Alcohol
⊞ Attire
⊟ Main Ingredient
☐ Rice
☐ Pasta
☐ Pork
☐ Chocolate
☐ Cheese
☐ Fruit
☐ Beef
☐ Chicken
☐ Peas
☐ Beans
Enter another Ingredient
Enter

1 A Slice of New York	412 Reviews *3443 Stevens Creek Boulevard* *San Jose, CA 95117* 408-247-5423
2 Adelita's Taqueria	90 Reviews *1896 Curtner Ave* *San Jose, CA 95124* 408-369-9905
3 Antipasto's by Derose Gourmet Meat Fish & Deli	69 Reviews *3454 McKee Rd* *San Jose, CA 95127* 408-251-5647
4 Bakery Mexico	63 Reviews *2811 Story Rd* *San Jose, CA 95127* 408-272-3838

Figure 36. Clear All Options

There is uniqueness in the restaurant review configurator. If the user selects constraints within the same category, the result seen is a set of restaurants that meets at least one of the constraints chosen. For example, if the constraint breakfast is chosen followed by lunch, the first result will contain only restaurants with breakfast meals and the latter will result in restaurants that server either lunch OR breakfast. If the user selects constraints from different categories, then the result is an AND of the constraints. For instance, taking the previous example, if the user further selects a constraint of Good For Kids in the General category, then the result seen will be containing restaurants with breakfast or lunch meals and Good For Kids. In form of an equation, this is how it looks like: result = (Breakfast OR Lunch) AND (Good For Kids).

Survey

In any application involving users, it is vital to have a usability study to make sure the usability requirements have been met. Once the application has been created it is time to refer back to the usability goals defined in the UI section earlier and verify that they have been met. By providing a survey, the designers are able to find out if the desired tasks are completed without error and if the purpose of the web application is clear. The first step in a usability study is to decide what needs to be evaluated.

Since, the first application was created to test the theory that was constructed, it was decided that the usability study would be solely on the second application. Having decided this, the next step was to choose what to test of the application. After observing the requirements and the theory, it was determined to test the location text field input, the simple constraints, and the ingredient constraint. The application would be compared to Yelp's, and so the study would also involve the users exploring that application as well. The users of the case study would be asked a series of questions which would then be gathered and analyzed. The users were students from San José State University. Because the application is created for a general audience a random set of users from the Student Union were selected for the case study. The survey was to see how user friendly the web application is and how accurate it is in terms of results for the users. The next section explains the case study.

Case Study

Because this web application is using a new innovative approach, anonymous surveys were conducted. The subjects used the web application which they accessed through my laptop or their own to answer the questions on the survey. The answers were based on their experience of using the web application. There were no risks in the subjects taking the survey. The only device needed was a computer. There was no obligation to which computer device the users were restricted to. Each user was approached by me. The project, the basic concepts, was explained. They were then asked if they would like to take an anonymous survey. Upon agreement, they were handed a consent form. Since it was anonymous, the consent form was of mine and the users did not need to sign it. The survey can be seen in Appendix B.

These users were students from the Student Union. The subjects were told in detail the project, the purpose of the survey, and the fact that only their answers will be analyzed and none of their personal information will be recorded. In fact, the survey did not ask any personal information about them. The users were not forced upon to take the survey as it was strictly voluntary. The results obtained are discussed below.

The Results

For the most part, the Restaurant Reviews application was liked. The tables below, table 6 and 7, summarize some of the results. All the users who took the survey felt that refining the restaurants was easy. This is shown in table 8. Table 9 shows the results of whether the users found the refining of Yelp.com easy. Four users did and six users did not find the refining easy at all.

Table 6 Question 1 Result

Yes	No
10	0

Table 7 Question 4 Result

Yes	No
4	6

Some of the general questions section results are shown below. The result to question 7, whether the Restaurant Reviews has better results and navigation than Yelp, show that the majority of the users found Restaurant Reviews much better than Yelp. A few of the users found the Restaurant Reviews application either somewhat better or about the same. One user did find it much worse. This user explained this choice in the comments section. Table 9 shows the results to the question of if the user would recommend Restaurant Reviews to others. Ninety percent of the users said they would while one user said the application would probably not be recommended. It was also necessary to know whether the users ever had experience searching for restaurants online. This aspect will affect their user experience. This is because even though only two applications are compared here, an experienced restaurant searcher will be comparing the applications to all the ones he or she has used. An inexperienced user will be practicing this type of search for the first time and will yield to different emotions. As a result, as seen in table 10, majority of the users had much experience in searching for restaurants online while three had somewhat experience and one did not have any experience at all.

Table 8 Question 7 Result

Much Better	**Somewhat Better**	**About the Same**	**Much Worse**	**NA**
6	1	2	1	

Table 9 Question 9 Result

Definitely Recommend	**Probably Recommend**	**Not Sure**	**Probably Not Recommend**	**Definitely Not Recommend**
6	3		1	

Table 10 Question 10 Result

Much Experience	**Somewhat Experience**	**No Experience**
6	3	1

The survey allowed for a way to see that the application was user-friendly and the purpose of the application was understood by the users. The comments section brought great constructive criticism. It was interesting to see users' point of view on the application and the way it made them feel. The comments and detailed results can be found in Appendix B.

Future Work and Conclusion

Future Work

Despite the application having many features and great layout, there are a couple items that would result in a greater user experience. The first item on the agenda is more restaurants. It can be researched into how this would be done. One idea is to go into each city's restaurants page on Yelp and attain a few key things to get the entire restaurant collection in a city. The first step would be to get the total number of restaurants in that city. With the use of a regular expression, this can be done. After this, the total restaurant number can be divided by ten so that the number of pages the restaurants will be in can be obtained. Yelp displays ten restaurants at a time and so by looking at the url, it can be modified p pages time and the HTML contents of that page can be obtained. The use of regular expressions can be used to obtain the desired data of each restaurant. Of course this is just one way of obtaining more restaurants. This technique does pose XML file size and parsing time issues. With more data, the XML file size will be more and so new ideas for storing the data efficiently in XML files will need to be thought of. All this parsing will be done only once, so the amount of time this will take needs to be considered. One must first test the idea on small data and then run it on the entire site.

The next item that needs to be improved is the display of cuisines. There are 82 cuisines, and displaying them only in one column becomes tedious to scroll through. A technique to provide the options in a horizontal pop-up needs to be thought of. This will reduce the space issue problem and provide a better user experience. Again the pop-up needs to be configured in a way that the AJAX technology is still working.

Since there are close to seven thousand cities the user can enter as the location, it would be nice to provide an autocomplete for that. This would require providing an array of the available cities. A more sophisticated autocomplete will be needed that not only takes cities whose first letter begins with the characters being entered, but also the second and third word beginnings. For example, if the user types the character L cities like Los Angeles and St. Louis should be in the values to choose from.

Conclusion

The Internet has evolved tremendously since its inception. From serving the military to the whole world, the Internet is now entering a new stage; customization. Web applications are now designed to give users a custom personal feel. Applications where the users ease of use and memorability of web applications are kept in mind have been changing the look and flow of web applications. Building products is the area where this is most prevalent. It uses the CSP as the backend and does not let the user realize what is happening in the background.

Earlier, the whole page needed to be refreshed if some sort of selection or change needed to be made in the page. Now with tools like AJAX, only areas of the web page that needs to be refreshed are and the rest remains the same. AJAX also allows the added capability of many AJAX calls in one method. The Main Ingredient text field constraint in the Restaurants Review application uses this feature. It first adds the constraint and sends it to the configurator. It then displays the ingredient name added to the checkbox list. Another new item presented in this paper is using only an XML, which has resulted in the elimination of a database.

The XML was the most important portion of the theory. Storing all the data needed for the application was quite a challenge. Once the format was finalized, coding the application was not nearly as tough. In addition, the use of SimpleXML in PHP and XPath allowed the finding of elements much easier and much more efficient than traversing through each node until the desired one is found.

Developing the event propagation theory for CSPs was not enough and the implementation of such a novel idea was required. The user selection defined an event that invoked the JavaScript method which began the solution propagation discussed. The use of AJAX solidified the fact that there are better ways of creating customized searches. This new way of creating quick, user-friendly applications is just the beginning. There is much more that can be done in this area.

Works Cited

"Ajax (programming)." Wikipedia, The Free Encyclopedia. 16 March 2009. Wikimedia Foundation, Inc. 16 March 2009. <http://en.wikipedia.org/wiki/AJAX>.

Ajax Tutorial. 2009. W3Schools. 1 March 2009 < http://www.w3schools.com/Ajax/Default.Asp>.

Apt, Krzysztof. Principles of Constraint Programming. Cambridge: Cambridge University Press, 2003.

Ardissono, Liliana, Er Felfernig, Gerhard Friedrich, Anna Goy, Dietmar Jannach, Giovanna Petrone, Ralph Schäfer, and Markus Zanker. "A framework for the development of personalized, distributed web-based configuration system." AI Magazine, 24 (2003): 93-110.

Castro, Elizabeth. XML For the World Wide Web. Berkeley: Peachpit Press, 2001.

"Constraint Programming" Wikipedia, The Free Encyclopedia. 13 March 2009. Wikimedia Foundation, Inc. 13 March 2009. <http://en.wikipedia.org/wiki/Constraint_programming>.

"Constraint Satisfaction Problem" Wikipedia, The Free Encyclopedia. 13 Feb. 2009. Wikimedia Foundation, Inc. 10 March 2009. <http://en.wikipedia.org/wiki/Constraint_satisfaction_problem>.

Constraint Solving 2.0 Ed. Martine Ceberio. 2009. 1 March 2009 <http://www.constraintsolving.com/>.

Goodman, L. "The Evolution of the Internet." FREEDOM. vol. 27, no. 4. 2004. 10 March 2009 <http://www.theta.com/goodman/evolve.htm>.

"Internet" Wikipedia, The Free Encyclopedia. 14 March 2009. Wikimedia Foundation. 14 March 2009 <http://en.wikipedia.org/wiki/Internet>.

J. Toussaint, K. C. "Web-based CBR (case-based reasoning) as a tool with the application to tooling selection." The International Journal of Advanced Manufacturing Technology. 29 (2006): 24–34.

Lewis, Bill. "Many-To-Many Relationships in XML Schema: Hints and Tips For the Busy E-R Modeler." The Data Administration Newsletter – TDAN.com. (2006). 14 March 2009 <http://www.tdan.com/view-articles/5046 >.

Link, Sebastian, and Thu Trinh. "Know your Limits: Enhanced XML Modeling with Cardinality Constraints." ACM International Conference Series. 334 (2007): 19-30.

Preece, Jennifer, Helen Sharp, and Yvonne Rogers. Interaction Design: Beyond Human-Computer Interaction. NY: John Wiley & Sons, Inc., 2002.

Rossi, F., P. Van Beek, and T. Walsh, eds. Handbook of Constraint Programming. UK: Elsevier, 2006.

Sabin, Daniel, and Rainer Weigel. "Product Configuration Frameworks-A Survey." IEEE Intelligent Systems. 13 (1998): 42-49.

Schmieg, S. "Evolution of the Internet." Weblog entry. You've Got Ismail! 19 Jan. 2006. 10 March 2009 <http://salimismail.com/?p=12>.

Stone, Debbie, Caroline Jarrett, Mark Woodrffe, and Shailey Minocha. User Interface Design and Evaluation. San Francisco: Morgan Kaufmann Publishers, 2005.

"WAMP." Wikipedia, The Free Encyclopedia. 10 March 2009. Wikimedia Foundation, Inc. 12 March 2009. <http://en.wikipedia.org/wiki/WAMP>.

XML: An Introduction – Brief History.RoseIndia. 10 March 2009 <http://www.roseindia.net/xml/xml-history.shtml>.

XML Representation of Constraint Networks Format XCSP 2.1. Organising Committee of the Third International Competition of CSP Solvers. (2008): 1-49.

XML to HTML. 2009. W3Schools. 1 March 2009 <http://www.w3schools.com/xml/xml_to_html.asp>.

Yelp. 2009. 10 March 2009 <http://www.yelp.com>.

APPENDIX A: Survey and Results

Event Based Propagation Approach to Constraint Configuration Problems

Survey

Please write your major:____________________

To answer the next set of questions, please follow the provided directions:

In your browser go to http://mh213d.cs.sjsu.edu/classes/cs299/rajhdeep/. You will see the web application Restaurant Reviews. Now in the text box, enter the city and state name as follows: San Jose, CA. Click the Search button. Look at the results. On the left side you will see a set of constraints you can narrow your results by. Select the meal type to be lunch and Cuisine to Mexican. You see that the result set has become smaller.

Now clear all the options you have selected and you can find restaurants with options of your own choice. Notice that in the Main Ingredient Constraint, you are able to type ingredient names not shown. Try entering more ingredients and see the results. After having done this answer the following questions:

1. Was refining the restaurants easy?
   i. Yes
   ii. No

Yes	No
10	0

2. How satisfied are you of the results?
   i. Extremely satisfied
   ii. Very satisfied
   iii. Neutral
   iv. Very dissatisfied
   v. Extremely dissatisfied

Extremely Satisfied	Very Satisfied	Neutral	Very Dissatisfied	Extremely Dissatified
2	4	4	0	0

3. How was the user experience of this web application?
   i. Extremely satisfying
   ii. Very satisfying
   iii. Neutral
   iv. Very dissatisfying
   v. Extremely dissatisfying

Extremely Satisfying	Very Satisfying	Neutral	Very Dissatisfying	Extremely Dissatisfying
4	5	1	0	0

Now go to www.yelp.com and in the text field type the same city and state name: San Jose, CA. Do not hit Enter of click on the Search button. Select the restaurants link underneath the browse by category section. This is located under the location information. You will then click on the more San Jose restaurants link located after the fifth restaurant. You will see that you can refine by the cuisine type. Click on Mexican here. You then see more filters. Click on More features and select Meals Served to be Lunch. See the results. You can also go back and choose options of your choice. Answer the following questions once done.

4. Was refining the restaurants easy?
   i. Yes
   ii. No

Yes	No
4	6

5. How satisfied are you of the results?
   i. Extremely satisfied
   ii. Very satisfied
   iii. Neutral
   iv. Very dissatisfied
   v. Extremely dissatisfied

Extremely Satisfied	Very Satisfied	Neutral	Very Dissatisfied	Extremely Dissatified
2	6	2	0	0

6. How was the user experience of this web application?
   i. Extremely satisfying
   ii. Very satisfying
   iii. Neutral
   iv. Very dissatisfying
   v. Extremely dissatisfying

Extremely Satisfying	**Very Satisfying**	**Neutral**	**Very Dissatisfying**	**Extremely Dissatisfying**
0	5	4	1	0

Questions in General:

7. Compared to yelp, would you say that Restaurant Reviews is __ in results and in navigation:
   i. Much better
   ii. Somewhat better
   iii. About the same
   iv. Much worse
   v. Don't know or never used

Much Better	**Somewhat Better**	**About the same**	**Much Worse**	**NA**
6	1	2	1	0

8. If given the chance, would you use Restaurant Reviews again?
   i. Definitely will
   ii. Probably will
   iii. Might or might not
   iv. Probably will not
   v. Definitely will not

Definitely will	**Probably will**	**Might not**	**Probably not**	**Definitely not**
4	5	1		

9. Would you recommend Restaurant Reviews to others?
   i. Definitely will recommend
   ii. Probably will recommend
   iii. Not sure
   iv. Probably will not recommend
   v. Definitely will not recommend

Definitely Recommend	Probably Recommend	Not Sure	Probably Not Recommend	Definitely Not Recommend
6	3	0	1	0

10. How much experience do you have in searching for restaurants online?
    i. Much experience
    ii. Somewhat experience
    iii. No experience

Much Experience	Somewhat Experience	No Experience
6	3	1

Any comments or suggestions on the Restaurant Reviews web application:

- "The Restaurant Review application is really good. The search is very simplified. It's easy to navigate through the application. If possible, Google maps can be used to make it easier for people to locate the restaurants on the city map."
- "One thing nice about Restaurant Reviews is in the ease of searching a place for lunch, breakfast etc. The pull down menu is very easy to use. In this respect it is better than yelp."
- "Restaurant Reviews seems to lag a second or so from when a constraint is added."

- "In regards to navigation and filtering the results, Restaurant Reviews (RR) was a much better experience as compared to Yelp. With Yelp, the filters could not be modified as easily as in RR. On the whole RR was a simple and quick experience for finding restaurants in a desired area."
- "Overall, a very good application to use. Narrowing down the restaurants was very easy and quick through the filter options."
- "Refining the results was easy; however, the layout of the filtering options was not user friendly. It requires a lot of scrolling to view all of the filtering options (reference to cuisine). I like how the results update without a page refresh, but there was a little delay before the results got updated. Maybe show a little hour-glass or animated gif when the results are updating. If I change the filter options, but the results do not change, there should be an indication that my filter options actually got applied. Maybe add a summarization of my filters above the results, and then I can quickly see all of the filters that have been applied. A much larger selection of restaurants should be added. I definitely like the Main Ingredient filter. The Rating filter looks for an exact matching number of stars, but I think if a user chooses 3 stars, then the results should include

any restaurant with 3 or more stars."

- "The one thing I noticed was the cuisine selection that is too large to be on the left panel. If you notice, the user has to scroll down for the lower alphabet country, on selection of the same, user cannot see the result until he again scrolls back up. Here I think the user might get confused that there is no result as it shows a blank white page. It may have been like yelp on the top of the results, the user might see the result below the selection. So it keeps the user engaged. The overall search distillation was really optimized and good to generate targeted results for the user. Great Work."

APPENDIX B: Glossary (courtesy of Wikipedia)

Advanced Research Projects Agency (ARPA): It is an agency of the United States Department of Defense creating new technology for the military.

Application Programming Interface (API): a set of routines, data structures, object classes and/or protocols provided by libraries and/or operating system services in order to support the building of applications.

Artificial Intelligence (AI): is the intelligence of machines and the branch of computer science which aims to create it.

Asynchronous JavaScript and XML (AJAX): Technique to build web applications

Cascading Style Sheet (CSS): is a style sheet language used to describe the presentation (that is, the look and formatting) of a document written in a markup language.

Case-based reasoning (CBR): is the process of solving new problems based on the solutions of similar past problems.

Computer-Aided Design (CAD): is the use of computer technology to aid in the design and particularly the drafting (technical drawing and engineering drawing) of a part or product, including entire buildings.

Computer Science (CS): is the study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.

Constraint Satisfaction Problem (CSP): mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations.

Direct Manipulation (DM): is a human-computer interaction style which involves continuous representation of objects of interest, and rapid, reversible, incremental actions and feedback.

Extensible Markup Language (XML): is a general-purpose specification for creating custom markup languages.

Extensible Stylesheet Language (XSL): a family of transformation languages, allows one to describe how to format or transform files encoded in the XML standard.

General Education (GE): set of general courses all university students must take to graduate.

HyperText Markup Language (HTML): predominant markup language for Web pages.

P2P: Peer-to-Peer file sharing is a method of distributing electronically stored information.

PHP: is a scripting language originally designed for producing dynamic web pages.

San Jose State University (SJSU): is the founding campus of what became the California State University system.

Standard Generalized Markup Language (SGML): is an ISO Standard metalanguage in which one can define markup languages for documents.

Stanford Research Institute (SRI): Established in 1946 it is located in Menlo Park, CA, United States.

University of California Los Angeles (UCLA): Founded in 1919 it is a public research university located in Westwood, Los Angeles, California, United States.

User Interface (UI): is the aggregate of means by which people, the users, interact with the system, a particular machine, device, computer program or other complex tool.

WAMP: an acronym formed from the initials of the operating system (Windows) and the package's principal components: Apache, MySQL and PHP (or Perl or Python).

World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web (abbreviated WWW or W3). It is arranged as a consortium where member organizations maintain full-time staff for the purpose of working together in the development of standards for the World Wide Web.

Appendix C: XML Snippets

XML Sample From Paper.

It supports configuration systems like CHOCO.

```
<instance>
 <presentation name = 'put here the instance name'
             maxConstraintArity = 'put here the greatest constraint arity'
             minViolatedConstraints = 'the minimum number of violated
                                        constraints'
             nbSolutions = 'put here the number of solutions'
             solution = 'put here a solution'
              type = 'CSP'
              format = 'XCSP 2.1'>
     Put here the description of the instance
 </presentation>

 <domains nbDomains='q'>
   <domain name = 'put here the domain name' nbValues = 'put here the
    number of values' >
     Put here the list of values
   </domain>
   ...
 </domains>

 <variables nbVariables='n'>
   <variable name = 'put here the variable name' domain = 'put here the name of
     a domain'/>
     ...
 </variables>

 <relations nbRelations='r'>
   <relation name = 'put here the name of the relation' arity = 'put here the arity
            of the relation'
            nbTuples = 'put here the number of tuples' semantics = 'put here
            either supports or conflicts' >
        Put here the list of tuples
      </relation>
      ...
    </relations>

    <predicates nbPredicates='p'>
      <predicate name = 'put here the name of the predicate' >
        <parameters>
            put here a list of formal parameters
        </parameters>
        <expression>
           Put here one (or more) representation of the predicate expression
        </expression>
      </predicate>
      ...
    </predicates>

    <constraints nbConstraints='e'>
      <constraint name = 'put here the name of the constraint' arity = 'put here the
           arity of the constraint'
           scope = 'put here the scope of the constraint'
           reference = 'put here the name of a relation, a predicate or a global
           constraint'>
        ...
      </constraint>
      ...
    </constraints>

  </instance>
```

Course Schema

```
<xsd:schema          ="http://www.w3.org/2001/XMLSchema"
                         ="qualified"                    ="unqualified">
<xsd:notation       ="Altova-CourseInformation"
              ="http://www.xmlspy.com/schemas/Altova/courseinformation"/>
<xsd:element       ="CourseInformation">
        <xsd:complexType>
                <xsd:sequence          ="3">
                        <xsd:element    ="CourseSet"/>
                </xsd:sequence>
                <xsd:attribute      ="degree"      ="xsd:string"      ="required"/>
        </xsd:complexType>
</xsd:element>
<xsd:element       ="CourseSet">
        <xsd:complexType>
                <xsd:sequence          ="unbounded">
                        <xsd:element    ="Course"/>
                </xsd:sequence>
                <xsd:attribute      ="name"      ="xsd:string"      ="required"/>
        </xsd:complexType>
</xsd:element>
<xsd:element       ="Course">
        <xsd:complexType>
                <xsd:attribute      ="courseName"      ="xsd:string"      ="required"/>
                <xsd:attribute      ="preReqs"      ="xsd:string"      ="required"/>
                <xsd:attribute      ="coReq"      ="xsd:string"      ="optional"/>
        </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Ingredient Association Schema

```
<xs:schema          ="http://www.w3.org/2001/XMLSchema"
                   ="qualified"                        ="unqualified">
<xs:element       ="Association">
        <xs:complexType>
                <xs:sequence>
                        <xs:element    ="Cuisine"            ="unbounded"/>
                        <xs:element    ="Ingredient"           ="unbounded"/>
                        <xs:element    ="Cuisine_Ingredient_Association"
                                                ="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element       ="Cuisine">
        <xs:complexType>
                <xs:sequence>
                        <xs:element      ="cName"      ="xs:string"/>
                        <xs:element      ="cID"      ="xs:string"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element       ="Ingredient">
        <xs:complexType>
                <xs:sequence>
                        <xs:element      ="iName"      ="xs:string"/>
                        <xs:element      ="iID"      ="xs:string"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element       ="Cuisine_Ingredient_Association">
        <xs:complexType>
                <xs:sequence           ="unbounded">
                        <xs:element      ="cuisineID"      ="xs:string"/>
                        <xs:element      ="ingredientID"      ="xs:string"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

Appendix D: Source Code Snippets

PHP Regular Expression to Attain All Cities from a State

Looking for pattern: <a href="/adamsville-al">Adamsville</a>

```
preg_match_all("!(<a href=\"\/[a-z\-]*[\-][a-z][a-z]\">)([A-Za-z ]*)(</a>)!",$html,
$cityMatches);
```

PHP Regular Expression to Attain Information About a Restaurant

Looking for pattern: <li><strong>Accepts Credit Cards:</strong> Yes</li>

```
preg_match_all("!(<li><strong>)([A-Za-z ]*:)(</strong>)([A-za-z ]*)(</li>)!",$html,
$restData);
```

PHP Script to Attain the Restaurants of a Particular Location

```
$searchLoc = $xml-
>xpath("/cityRestaurants/city[@cityName='$cityName'][@cityState='$cityState']");
```

Appendix E: Cuisine and Ingredient Names

Cuisine Names

Afghan	Gluten-Free	Spanish
African	Greek	Steakhouses
American (New)	Halal	Sushi Bars
American (Traditional)	Hawaiian	Taiwanese
Argentine	Himalayan/Nepalese	Tapas Bars
Asian Fusion	Hot Dogs	Tex-Mex
Barbeque	Hungarian	Thai
Basque	Indian	Turkish
Belgian	Indonesian	Ukrainian
Brasseries	Irish	Vegan
Brazilian	Italian	Vegetarian
Breakfast and Brunch	Japanese	Vietnamese
British	Korean	
Buffets	Kosher	
Burgers	Latin American	
Burmese	Live/Raw Food	
Cajun/Creole	Malaysian	
Cambodian	Mediterranean	
Caribbean	Mexican	
Chicken Wings	Middle Eastern	
Chinese	Modern Europe	
Dim Sum	Mongolian	
Creperies	Moroccan	
Cuban	Pakistani	
Delis	Persian/Iranian	
Diners	Pizza	
Ethiopian	Polish	
Fast Food	Portuguese	
Filipino	Russian	
Fish and Chips	Sandwiches	
Fondue	Scandinavian	
Food Stands	Seafood	
French	Singaporean	
Gastropubs	Soul Food	
German	Southern	

Ingredient Names

Almond	cherry	green beans	noodle
american cheese	chicken	guava	nutmeg
Anchovy	chickpeas	ham	nuts
Antelope	chilli	hash brown	oats
Apple	chocolate	hemp seed	oil
Apricot	cilantro	herbs	okra
Artichoke	cinnamon	honey	olive
Asparagus	clams	horseraddish	olive oil
Avocado	coconut milk	jackfruit	onion
Bacon	cod	jalapeno	orange
Banana	coffee	jam	oregano
Barley	corn	jelly	ox
Basil	cottage cheese	kebabs	oyster
bay leaf	crab	ketchup	paneer
Beans	crawfish	kimchi	papaya
Beef	cream cheese	lamb	paprika
Beetroot	crocodile	leek	parmesan
bell pepper	cucumber	lemon	parsley
Berries	cumin	lemongrass	parsnips
Bison	dog	lentil	pasta
Blueberry	dressing	lettuce	peach
bok choy	duck	lima	peanut
Bologna	durian	lime	peanut butter
Bread	eggplant	lobster	pear
Breadcrumbs	eggs	longan	peas
Breadfruit	figs	lychee	pesto
Broccoli	fish	mango	pig
brown sugar	fish sauce	mangosteen	pineapple
Butter	fruit	mayonnaise	plantain
Cabbage	fudge	meatball	plum
Caramel	garam masala	melon	pomegranate
Cardamom	garlic	milk	pork
Carp	ghee	millet	potato
Carrot	ginger	mint	prawns
Catfish	ginseng	monkey	provolone
Cauliflower	goat	mozzarella	prune
Celery	goat cheese	mushroom	pumpkin
cheddar cheese	goose	mussels	quince
Cheese	grapes	mustard	rabbit

Raisin	tomato
Rambutan	tortilla
Ranch	trout
Raspberry	tuna
refried beans	turkey
Relish	turmeric
Rice	turnip
Rye	vanilla
Saffron	veal
Salami	vermicelli
Sambal	vinegar
Sausage	wasabi
Scallions	water chestnut
Seafood	watermelon
Seaweed	wheat
Sesame	whip cream
Sheep	wine
Shrimp	yam
sour cream	yogurt
soy sauce	zucchini
Soybeans	
Spam	
Spinach	
Sprouts	
Squash	
Squid	
Steak	
Strawberry	
Sumac	
Sushi	
sweet potato	
Syrup	
Tamarind	
Tapioca	
Taro	
Tempe	
Teriyaki	
Thyme	
Tofu	