IS 257 Database Management
Final Project Report

John Fritch, Diane Ghorbani, Leticia Valdez
December 5, 2002
Table of Contents

Summary 3
Application Overview 4
Application Description 6
Queries, Forms, and Reports 13
Appendices 25
Summary

Group Voice is designed to allow group leaders to gather the opinions of their members, in order to make decisions on policy matters. Unlike other tools available today, such as email, survey tools and traditional decision support systems, the focus of Group Voice is the presentation and summarization of opinions of members, on questions proposed by the leaders. In addition, Group Voice archives each question’s supporting documentation, opinions and final outcome for future use by the group.

In partnership with the Latino Issues Forum, a group of SIMS Masters students has developed the first prototype of Group Voice. The system provides an effective means of collecting the opinions of a group who cannot easily gather at a centralized point and time. This web-based collaborative tool allows members of LIF to access a question posed by leadership at a time of their choosing. The members are able to review the question, supporting documentation, options and arguments submitted. They may then concur with arguments, which express their opinions, and add new arguments to further explain their position on the question. Leaders will monitor levels of participation on each question as well as the levels of support for the various options. Once the time period for comment has concluded, leaders may then post their decision regarding the question.

The application prototype is built on a Microsoft Access database using a ColdFusion application server and a Microsoft IIS web server. The tables within the database are in three main categories: participants, questions, and opinions. The application views incorporate information from all three categories to present Participants, Leaders, and Administrators with information pertinent to them.

The database design proved very efficient for data input. However, there were challenges in working with the highly normalized tables, particularly in presenting summarized information to leaders. To overcome this challenge, database views were created which combined data from multiple tables. Summary queries then referenced these database views.
Application Overview

It is generally the responsibility of the leaders to make decisions on issues of interest to their group. This decision-making is not usually done in a vacuum, but rather involves obtaining the opinions of group members. When the group is unable to meet at a central location and time to discuss issues, it becomes challenging for the leaders to gather the members’ opinions. Many groups try to manage this process via email or electronic bulletin boards where various parties present their opinions in numerous discussion threads. These processes provide the opportunity for opinions to be overlooked in a myriad of responses, as well as leaving leaders with the time-consuming process of accumulating and summarizing the results. Furthermore, these processes are not well suited to centralizing the documentation used to arrive at a decision and maintaining a history of the decision-making process.

Group Voice addresses these issues via a centralized web application that enables an organization to:

- Present questions
- Collect the opinions
- Provide leaders with a summary of the group’s opinions
- Archive questions and opinions for future reference

Market

Group Voice is designed to meet the needs of groups in which the leaders make decisions on policy-related matters for the group. The leaders of these groups look to membership for insight on particular factual data as well as their opinions of the value of taking a specific position on the issues under review. Leaders also try to engage new members in the process to help them become familiar with the approach the members use to express their opinions.

Groups which are large in number and/or not centrally located find it difficult to meet at a central point and time to provide input. Thus they rely on tools such as Email, Message Boards, Survey Tools and traditional Decision Support Systems to gather input.

First Customer

Our first customer is the Latino Issues Forum (LIF)

LIF is a non-profit public policy and advocacy institute dedicated to advancing new and innovative public policy solutions for a better, more equitable and prosperous society. Established in 1987, LIF’s primary focus is on the broader issues of access to higher education, economic development, health care, citizenship, regional development, telecommunications issues and regulatory issues. LIF also serves as a clearinghouse to assist and provide the news media with accurate information and sources in Latino
community for fair and effective coverage of issues. LIF addresses public policy issues from the perspective of how they will affect the social and economic future of the Latino community.¹

LIF has three directors with decision-making responsibility and approximately 20 members, who are located at three sites in California. The directors like to get input from staff when making decisions, however it is difficult to get staff members together at the same place and time. They have indicated that some of their best discussions occur during semi-annual retreats. During retreats, all members of the organization have a chance to provide opinions on issues of interest to LIF. However, these retreats require a lot of advance planning and only occur twice a year.

In between retreats, the leaders request input via email or one-on-one discussions with knowledgeable members. Neither method has proved to be as fruitful as the retreat discussions. LIF leadership would like to create a forum for discussion that is not dependent upon the group coming together.

**Terminology**

In order for leaders to gather the input of their members, they must clearly outline the question. For purposes of Group Voice, a **question** is the issue on which the group is being asked to provide opinions. When asking for the input of the group, the leader offers various **options** or responses to the question. Members are asked to provide **arguments** in favor of the options they support. Should members be uncertain about the question, they may pose a question back to the leadership in the form of a **clarification**. When leaders and members wish to refer to specific support for their arguments, they may submit **documentation** in the form of a file, a URL link or a bibliographic reference. When members decide that existing arguments support their beliefs, they may **concur** with the arguments. Once the opinions of members have been collected, leaders decide which position the group will take on the question, this **outcome** is then posted in Group Voice for all members to view. The bold terms are used throughout the design of Group Voice as well as discussions of its functionality below.

¹ [http://www.lif.org/about.html](http://www.lif.org/about.html) accessed December 1, 2002.
Application Description

Group Voice is intended for groups in which leadership solicits opinions from a distributed membership.

Group Voice is web-based collaborative tool to:

- Present questions (with options and documentation) to a group
- Collect the opinions (expressed as arguments and concurrences) in favor of various options
- Provide leaders with a summary of the group’s opinions
- Archive questions and opinions for future reference

Group Voice contains three views: participant, leader and administrator. The participant view lists the questions (with related options and supporting documentation). The leader view allows quick review of participation levels and levels of support for the various proposed options. The administrator view lists the questions the administrator is managing.

Participants have the following functionality for questions they have been invited to:

- See questions, options and arguments supporting each option.
- Review documentation supporting questions, options and arguments.
- Concur with an argument supporting a proposed option.
- Provide a new argument.
- Provide documentation related to the question.
- Review resolved questions, options, arguments and supporting documentation as well as the outcome.

Leaders have the following options:

- Review current levels of participation for open questions.
- Review current levels of support for various options both for mandatory and optional participants.
- Review historical participation levels on closed questions.

Administrators have the following options:

- Set up a new question by providing the questions title, detailed description, options, and documentation.
- Invite participants to participate in the question.
- Post the outcome for a question once a decision has been made by the leadership.

The initial design focused on only two views (participant and administrator). Following conversations with LIF, it became clear that a view for the leadership is critical. In
addition, LIF considered the archiving of old questions and supporting information to be important. The group leadership felt this was particularly useful in training their internship staff, on how to evaluate a question. Both of these functional areas have been expanded in the current version of Group Voice.

**Architecture**

For the first prototype, the database is Microsoft Access; the application server is ColdFusion; and the web server is Microsoft IIS. These components facilitated the rapid development of the prototype.

There are three types of tables within the database: participant tables, question tables and opinion tables. The participant tables include information such as email address, password and status (leader, participant, administrator). The question tables include information such as the options, supporting documentation and close date. The opinion tables incorporate information regarding the arguments and concurrences provided by participants.

The interface was created using a combination of ColdFusion and HTML. ColdFusion allows for dynamic creation of all pages within the application, based upon information maintained in the database.

This architecture is revised from the original concept to facilitate rapid development of the first prototype. Once the prototype is tested and revised, the final architecture will be based on LIF’s preferences and existing infrastructure.

**Database**

As previously mentioned the database consists of three types of tables: participant, question and opinion.

Information regarding participants includes their email address (used as the user id of log in purposes) and their password. The unique Part_ID is used throughout the Group Voice application as it allows the determination of each participant’s level of functionality. Participants can be invited to a question, they may also be a question administrator as well as a leader. The information presented to them, within the application is derived based upon the way they have been set up within the database.

The question tables track information regarding the status of each question from newly created (New), available to participants (Posted) and no longer available for discussion (Closed). The Question_ID is used throughout Group Voice to track options, arguments, and documentation. In addition, the combination of the Question_ID and the Part_ID determines which questions are available to each participant.
The opinion tables track the creation of arguments and concurrences by participants. The concurrence information is key in developing statistics about levels of participation and support.

The highly normalized nature of the database is very useful for data entry. However, when attempting to accumulate statistics about participation and support for options, the queries become quite complex and difficult to debug. To facilitate easier querying, database views were created to accumulate Question_ID, Option_ID, Argument_ID, and Concurrence_ID so that a single view could be joined with the Part_Status table.

Currently, the database includes an inactive clarification table. This table will be used to implement future functionality.

A complete Relationship Diagram is available in Appendix A. The Data Dictionary is available as Appendix B.

**User Views**

ColdFusion allows for the dynamic creation of all pages, based upon the Part_ID and Question status. The actual workflow for each user is dependent upon their role (participant, leader, administrator). An overview of each of these workflows is presented here, and detail of the queries and forms presented later in this report will refer to these flows.
After logging in, participants see a page with all of the questions, both open and closed, that they have been invited to view.
Leaders are presented with the participant view first, and select Leader from the navigation bar, to view summarized participation on open and closed questions.
Administrators are presented with the participant view first, and select Admin from the navigation bar, to view the status of questions they are administering.
Features not supported within ColdFusion

The first prototype of Group Voice requires external maintenance of the database for several services. These services will be performed by a System Administrator, who is responsible for the creation of the database, as well as the establishment of the application environment. SIMS students currently serve in this role. The services maintained externally include:

- Creation of new participants – new participant set up requires the entry of an email address, password, leader status and a unique Part_ID is generated.
- Creation of a new question – the initial set up of a question is done directly to the database. The system administrator enters a brief question_header and a unique Question_ID is generated. The status of the question is set to “New” as part of this set up.
- Set up of question administrator – the Part_ID and Question_ID are entered into the Question_Admin table, to allow the administrator to enter the specifics for the question.
- Archiving old questions – will be based upon volumes and specific organizational requirements for historical data.

Future inclusion of these features will depend upon the requirements addressed in user testing.
Queries, forms and reports

Each of the views (Participant, Leader and Administrator) of Group Voice includes queries, forms and reports. For purposes of this report extensive detail of the Participant View is provided below. In addition, some of the queries which are unique to the Leader View are presented. The Administrator view incorporates many of the same functionalities provided through the other views, and has not been included.

The full code for each bolded ColdFusion page is available in Appendix C.

All users log in to Group Voice from the same Login.cfm.

The page calls Validate.cfm which runs the following query:

```cftml
<CFQUERY NAME="ValidateUser" DATASOURCE="#Application.DSN#">
  SELECT Part_ID, Part_Name, Part_Email, Part_Password, Leader
  FROM Participant
  WHERE Part_Email = '#Form.Username#' AND Part_Password = '#Form.Password#'
</CFQUERY>
```
After logging in, participants see the Index.cfm page which contains several queries listing information specific to the participant.

**Participant Queries**

Welcome to the Group Voice questions for Latinx Issues Forum. U.I. is very interested in what you have to say about the posted questions and the possible options that U.I. can explore. Please post arguments in favor of the options as reasons that you believe that course of action is a good idea. If you agree with an argument that has been posted, simply post your concurrence with the argument.

You have been invited to participate in the following questions.

<table>
<thead>
<tr>
<th>Category</th>
<th>ID</th>
<th>Question</th>
<th>Close Date</th>
<th>Options</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Civic Participation</td>
<td>6</td>
<td>Should U.I. support Flag 42 (same-day voter registration)?</td>
<td>12/10/02</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Civic Participation</td>
<td>3</td>
<td>Should U.I. support Flag 427</td>
<td>12/15/02</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

**Select an Open Question:**

Click an open question to review the options and supporting arguments provided for this question.

**Review a Closed Question:**

Click a closed question to review the options and supporting arguments provided for this question.

<table>
<thead>
<tr>
<th>Category</th>
<th>ID</th>
<th>Question</th>
<th>Close Date</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sustainable Development</td>
<td>1</td>
<td>Should U.I. support Flag 402</td>
<td>10/20/02</td>
<td>Yes</td>
</tr>
<tr>
<td>Energy</td>
<td>2</td>
<td>Should U.I. support Flag 427</td>
<td>10/24/02</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Copyright 2000 Kirch, Glean, Merida. All Rights Reserved.
Query 1: List open questions for specific participant and sort by user selected parameter.

```cfml
<cfquery name="PostedQuestions" datasource="#Application.DSN#" >
  select Question.Question_ID, Question.Question_Header, Question.Close_Date,
  Question.Category_ID, Category.Category_Text
  from Question, Category, Part_Status
  where Question.Status_ID = 3 and Question.Category_ID = Category.Category_ID
  and Part_Status.Part_ID = #Session.Part_ID#
  and Question.Question_ID = Part_Status.Question_ID
  and Question.Close_Date > #now()#
  <cfif isdefined ("SortParam")>
    order by #SortParam#
  </cfif>
</cfquery>
```

Query 2: List the total number of options for a question from _CountOpts.cfm

```cfml
<cfquery name="OptionCount" datasource="#Application.DSN#" >
  select count(Option_ID) as TotalOptions
  from Question_Option
  where Question_Option.Question_ID = #Session.Question_ID#
</cfquery>
```

Query 3: List total number of arguments for all options for a question from _CountArgs.cfm.

```cfml
<cfquery name="ArgCount" datasource="#Application.DSN#" >
  select count(Argument.Argument_ID) as TotalArguments
  from Argument, Question_Option
  where Question_Option.Question_ID = #Session.Question_ID#
  and Question_Option.Option_ID = Argument.Option_ID
</cfquery>
```

Query 4: List closed questions for specific participant and sort by user selected parameter.

```cfml
<cfquery name="ClosedQuestions" datasource="#Application.DSN#" >
  select Question.Question_ID, Question.Question_Header, Question.Close_Date,
  Question.Outcome, Question.Category_ID, Category.Category_Text
  from Question, Category, Part_Status
  where Question.Status_ID = 4
  and Question.Category_ID = Category.Category_ID
  and Part_Status.Part_ID = #Session.Part_ID#
  and Question.Question_ID = Part_Status.Question_ID
  <cfif isdefined ("SortParam")>
    order by #SortParam#
  </cfif>
</cfquery>
```
Once participant selects an open question, the ParticipantQuestion.cfm presents the related question detail.

### Category: Time Participation

Should LIF support Prop S2 (same day voter registration)?

Prop S2 would allow persons who are legally eligible to vote and have valid identification to register to vote on election day at their polling place. Increases criminal penalty for voter and voter registration fraud. Criminalizes conspiracy to commit voter fraud. Requires trained staff at polling places to manage election day registration, creates funds to implement measures, including training and providing personnel for election day registration. Allows persons to register or re-register during 30 days preceding election day at local election offices. Provides more time for county election officials to process voter registration items. Annual state costs of about $20 million to fund counties for election day voter registration activities. Nationwide administrative costs are unknown, but probably minor. State costs to enforce a new election fraud offense.

Close 12/19/02 Participants: 18

<table>
<thead>
<tr>
<th>New</th>
<th>ID</th>
<th>Option</th>
<th>Number of Arguments</th>
<th>Number of Concurs</th>
<th>Most Recent Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>-</td>
<td>We should support proposition S2</td>
<td>2</td>
<td>7</td>
<td>11/07/02</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>No position. Proposition S2 does not pertain to LIF</td>
<td>2</td>
<td>8</td>
<td>11/08/02</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>Yes</td>
<td>3</td>
<td>21</td>
<td>12/02/02</td>
</tr>
</tbody>
</table>
The information in the question header is similar to Query 1 (above). In addition, a separate query which counts the total number of participants who have concurred with or created new arguments for the question (_CountParticipants.cfm).

Query 5: List each option and its respective number of arguments.

```cfquery
<cfquery name="OptionStatistics" datasource="#Application.DSN#">
  SELECT Question_Option.Option_Header, Question_Option.Option_ID, Question_Option.Option_Desc, count(Argument_ID) as TotArgs
  FROM Question_Option LEFT OUTER JOIN Argument ON Question_Option.Option_ID = Argument.Option_ID
  WHERE Question_ID = #Session.Question_ID#
  GROUP BY Question_Option.Option_Header, Question_Option.Option_ID, Question_Option.Option_Desc
</cfquery>
```

Query 6 List the available documentation for selected question from _Documentation.cfm.

```cfquery
<cfquery name="QuestionDocumentation" datasource="#Application.DSN#">
  SELECT Doc_Header, Doc_Type, Doc_Link, Doc_Date
  FROM Documentation
  WHERE Question_ID = #Session.Question_ID#
  ORDER BY Doc_Date
</cfquery>
```

Query 7: Calculate the number of concurrences for a for specific option from _OptionConcurCount.cfm.

```cfquery
<cfquery name="OptionConcurCount" datasource="#Application.DSN#">
  SELECT Count(Concurrence.Concur_ID) As Total
  FROM Concurrence, Argument
  WHERE Argument.Option_ID = #Session.Option_ID#
    AND Argument.Argument_ID = Concurrence.Argument_ID
</cfquery>
```

Query 8: List the date of the most recently submitted argument for an option from LatestArgDate.cfm.

```cfquery
<cfquery name="LatestArgDate" datasource="#Application.DSN#">
  SELECT max(Argument_Date) As LatestArgDate
  FROM Argument
  WHERE Option_ID = #Session.Option_ID#
</cfquery>
```
When a participant selects an option, the ParticipantOption.cfm presents the arguments in favor of the specific option.
Query 9: List the arguments the participant has not concurred with and sort by user selected parameter from _ArgumentsNoConcur.cfm.

<CFQUERY name="OptionArguments" datasource="#Application.DSN#">
    SELECT Argument_Header, Argument_ID, Argument_Desc, Argument_Date, 
    Participant.Part_Name, Count 
    FROM Argument_With_Count, Participant 
    WHERE Option_ID = #Session.Option_ID# 
    AND Author_ID = Participant.Part_ID 
    AND Argument_ID NOT IN 
        (SELECT Concurrence.Argument_ID 
         FROM Concurrence, Argument 
         WHERE Argument.Option_ID = #Session.Option_ID# 
         AND Concurrence.Argument_ID = Argument.Argument_ID 
         AND Concurrence.Part_ID = #Session.Part_ID#)

    <CFIF IsDefined("Session.SortParam")> 
        ORDER BY #Session.SortParam# 
    <CFELSE> 
        ORDER BY Argument_Date DESC 
    </CFIF>
</CFQUERY>

Query 10: Allow participant to concur with argument from _ShowConcur.cfm.

<CFQUERY name="ShowConcur" datasource="#Application.DSN#">
    SELECT Count(Concurrence.Concur_ID) As Total 
    FROM Concurrence 
    WHERE Concurrence.Argument_ID = #Session.Argument_ID# 
    AND Concurrence.Part_ID = #Session.Part_ID# 
</CFQUERY>

Query 11: Update concurrence table from ConfirmConcur.cfm.

<CFIF Form.ConcurStatus IS 1>
    <CFQUERY name="DeleteConcur" datasource="#Application.DSN#">
        DELETE FROM Concurrence 
        WHERE Concurrence.Part_ID = #Session.Part_ID# 
        AND Concurrence.Argument_ID = #Form.ConcurArg# 
    </CFQUERY>
    <CFELSE>
        <CFQUERY name="AddConcur" datasource="#Application.DSN#">
            INSERT INTO Concurrence (Concur_Date, Part_ID, Argument_ID) 
            VALUES ('#DateFormat(Now(), 'mm/dd/yyyy')#', #Session.Part_ID#, 
            Form.ConcurArg#) 
        </CFQUERY>
    </CFELSE>
</CFIF>
If a participant chooses to add an argument the **EnterArgument.cfm** wizard is displayed.

Query 12: Insert new argument in to the Argument table and insert a concurrence for that argument into the Concurrence table from **AddArgument.cfm**.

```cfml
<CFQUERY name="AddArgument" datasource = "#Application.DSN#">
  INSERT INTO Argument (Option_ID, Author_ID, Argument_Header, Argument_Desc, Argument_Date)
  VALUES (#Session.Option_ID#, #Session.Part_ID#,'#Form.Argument_Header#', '#Form.Argument_Desc#', '#DateFormat(Now(), 'mm/dd/yyyy')#')
</CFQUERY>

<CFQUERY name="GetMaxArgID" datasource = "#Application.DSN#">
  SELECT max(Argument_ID) As Max
  FROM Argument
</CFQUERY>

<CFSET MaxArgID = "#GetMaxArgID.Max#">
<CFQUERY name="ProvideConcur" datasource = "#Application.DSN#">
</CFQUERY>
```
INSERT INTO Concurrence (Concur_Date, Part_ID, Argument_ID)
VALUES (#DateFormat(Now(), 'mm/dd/yyyy')#, #Session Part_ID#, #MaxArgID#);
</CFQUERY>

<CFQUERY name="AddDocument" datasource = "#Application.DSN#">
  INSERT INTO Documentation (Doc_Link, Doc_Type, Question_ID, Doc_Header, Part_ID, Doc_Date)
  VALUES ('#CFFILE.ClientFile#', 2, #Session.Question_ID#, '#Form.Doc_Header#', #Session.Part_ID#, #DateFormat(Now(), 'mm/dd/yyyy')#);
</CFQUERY>

(Queries also exist to insert url and citation information.)
Leader Queries

Once a leader has selected a specific question, the **LeaderQuestion.cfm** page is presented.
Query 1: Calculate the percentage of total concurrences that each option has garnered from _OptionConcurPercentage.cfm.

<CFQUERY name="OptionConcurCount" datasource="#Application.DSN#">
    SELECT Count(Concurrence.Concur_ID) As Total
    FROM Concurrence, Argument
    WHERE Argument.Option_ID = #Session.Option_ID#
    AND Argument.Argument_ID = Concurrence.Argument_ID
;</CFQUERY>

<CFQUERY name="QuestionConcurCount" datasource="#Application.DSN#">
    SELECT Count(Concurrence.Concur_ID) As Total
    FROM Concurrence, Argument, Question_Option
    WHERE Question_Option.Question_ID = #Session.Question_ID#
    AND Question_Option.Option_ID = Argument.Option_ID
    AND Argument.Argument_ID = Concurrence.Argument_ID
;</CFQUERY>

<CFIF QuestionConcurCount.Total IS NOT 0>
    #NumberFormat((OptionConcurCount.Total/QuestionConcurCount.Total*100), '999')##%
<CFELSE>NA
</CFIF>

Query 2: Calculate the total number of concurrences for each option and report the information in a bar graph from _graph.cfm

<CFQUERY name="QuestionConcurCount" datasource="#Application.DSN#">
    SELECT Question_Option.Option_Header, Count(Concurrence.Concur_ID) As Total
    FROM Concurrence, Argument, Question_Option
    WHERE Question_Option.Question_ID = #Session.Question_ID#
    AND Question_Option.Option_ID = Argument.Option_ID
    AND Argument.Argument_ID = Concurrence.Argument_ID
    GROUP BY Question_Option.Option_Header
</CFQUERY>

<CFIF QuestionConcurCount.RecordCount IS 0>
    No participation to graph.
<CFELSE>
    <CFGRAPH TYPE="HorizontalBar" QUERY="QuestionConcurCount" GRAPHHEIGHT="350" GRAPHWIDTH="350" VALUECOLUMN="Total" ITEMCOLUMN="Option_Header" ITEMLABELORIENTATION="Horizontal" SHOWNVALUELABEL="Yes" TITLE="Total Concurrences for each Option">
    </CFGRAPH>
</CFIF>
Query 3: Calculate the total number of concurrences by mandatory participants for each option and report the information in a bar graph from _graphMandatory.cfm.

(Note: This query involves the use of the view Concur_Details. See Appendix D.)

<CFQUERY name="QuestionMandatory" datasource="#Application.DSN#">
SELECT Question_Option.Option_Header, Sum(CountOfConcur_ID) as Total
FROM Question_Option, Concur_Details
WHERE Concur_Details.Question_ID = #Session.Question_ID# AND
Question_Option.Option_ID = Concur_Details.Option_ID AND Mandatory = 1
GROUP BY Option_Header
;
</CFQUERY>

(A similar query exists for optional participants in _graphOptional.cfm)
Appendices

A. Data Dictionary
B. Relationships Diagram
C. ColdFusion Templates
D. SQL Statements for Database Views