BEAR/Advisor

A System for Online Retreval of Registrar Information and Course Management

Version 1.01

Top Level Design Document

1.0 Document Overview

This document contains a top level design for the BEAR, the Brown Electronic Access to the Registrar and an accompanying Advisor. This system will allow for completing tasks involving the registrar such as dropping and adding courses via the World Wide Web. This document contains various top level components of design as well as annotations detailing the functionality of these components, and a timeline for the creating and implementing these components.

1.1 Top Level Components

At the highest level of this program, there are two main components of this project: the server and the client. The server will handle the secure storage of information and facilitate for similarly secure access to this information by students. This is where personal schedules are stored, course information, registration, grades, etc. will be stored. The user can then retreive this information usind any compatible client through this server. The client in this case will provide the user an interface with which to access information from the system via the World Wide Web.

2.0 Server Design

The server is designed to store information that needs to be accessed by the system. Its main purpose is to give access to information in a single,central location where it can be queried by the client. This is a visual representation of the server design:

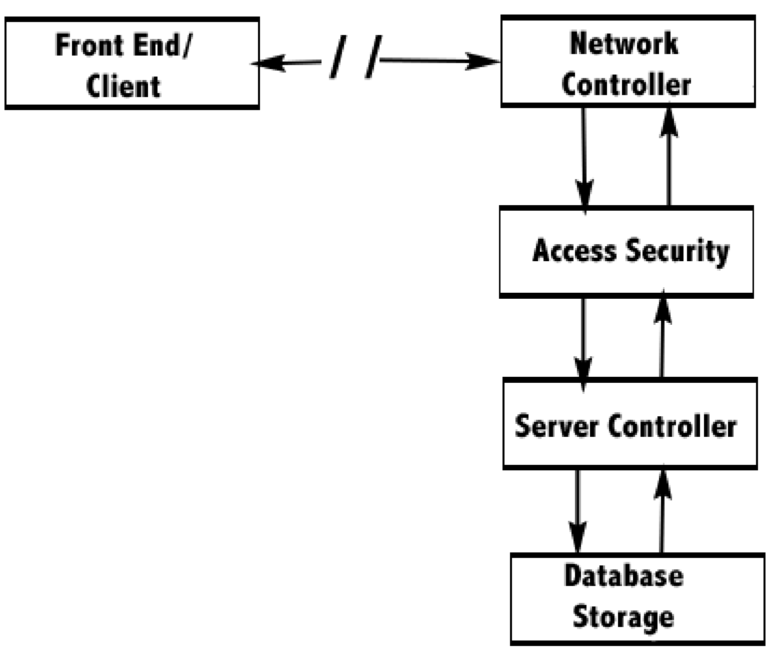

figure 1. Top Level Server Design

The **Network Controller** is responsible for controlling network via the clients. The main purpose for this aspect of the project is to encapsulate the rest of the server components below such that the client is unaffected by any changes internal to the server. A new Network Controller will exist for each client connection.

Access Security deals with all issues pertaining to data access. This is where protections on who can read/write specific data will be enforced. It is also responsible for implementing the inital user authentication, and providing a firewall to the information contained in the server below.

The **Server Controlle**r both translates encryped client requests into SQL database queries and re-translates the information returned from the query into a decripted reply. This information as it flows into and out of the Main Server must be standardized (therefore we must create some sort of language/rubric for encoding/decoding this information). The Server Controller also is responsible for finding what information is needed from the database, and from that where in the database to find that.

Database Storage basically just stores the infomation that the client will be trying to retreive. This will be a rather large database of information such as courses, grades, etc.

3.0 Client Design

The front-end of the system will provide an HTML interface between the user and the server. Its design is as such:

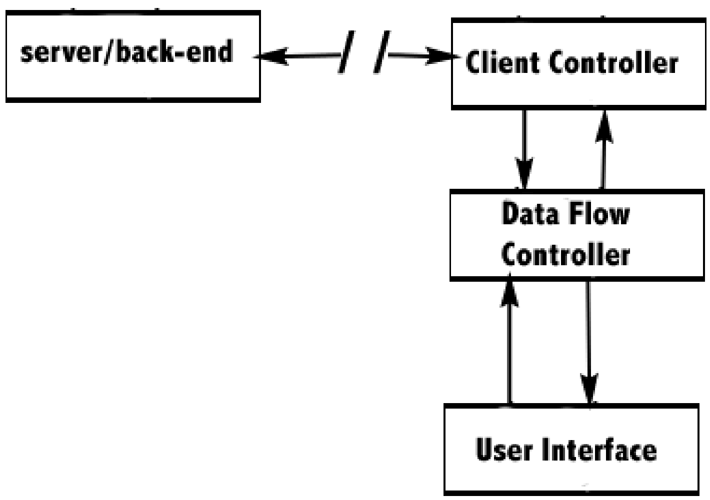

figure 2. Top Level Client Design

The **Client Controller**, like the Network Controller, is responsible for dealing with the transfer of information over the network. All communication to and from the server is filtered through the Client Controller. This will allow for us to use a common way of dealing with the server no matter who the client is. Each client will have its own Client Controller. There should also be some sort of data encryption/decryption here to increace security. Also like the Network Controller, the Client Controller allows for changes to be made to the client while not effecting the server side of the system at all.

The **Dataflow Controller** serves as an interface between the user input/output and the underlying client applications. It will serve as data traffic controller for clients who are requesting and receiving information from the server.

The **User Interface** is pretty self-explanitory. In the form of html pages, the User Interface will provide a vehicle for the user to get/receive information in a usable and readable format. See the specifications document for more in-depth representations of the UI.

4.0 External Dependencies

As far as external dependencies go there are a couple. Firstly, because this is a web based program, we are limited somewhat by the issue of accessability. Therefore we are bound by the necessity, as the specs lay out, of speed (no longer than 15 secs. to load over 28.8 mdem), cross-browser capability (usable on Netcape 3.0+ and IE 3.0+), and reliabilility. Secondly, there is the issue of actual database access. If we were to use the acutal server utlilzed by the registrar, our system would be relient upon the accessiblity of that database. However, as it appears that we will be using a mock database, this should not be an issue.

5.0 Task Breakdown/Group Organization

As I see it, this system has five main components with respect to actual coding: security, client, server, database handling, user interface. Therefore, these will be the divisions amongst the coding team members. In addition to this, there will need to be people responsible for documentation, testing, writing the manual, organizing the group, and leadership.

Therefore, loosely following the "Surgical Team" model suggested by Brooks in *Mytical Man Month* , I would suggest the following breakdown for team structure (potential team members who could fill these roles are in parentheses) :

Team Leader: Responsible for overseeing the project. He/she must know the program in and out so that he/she can properly advise/inform otehr team members of issues/problems that are occurring in the implementation process. (Joe)

Vice-team leader: Serves function similar to that of the vice-president. He/she must step in when and where ever is needed should the team leader not be available or if it is an area which does not require the leader's input. The vice-team leader also serves as a sort of arbitrator should disagreements, miscommunications, or problems arise between team members. (Melissa)

Documenter/Manual Writer: Because this person is documenting the overall program and writing the user manual, he/she must know nearly as much about the project as the two leaders. Therefore in addition to doing these two tasks, this person should also serve as a reference for the coders to ask various questions about implementation. Should both leaders be occupied, then team leader responsibilities should fall on this person. Also, this team member is responsible for maintaining a team web page that will provide a forum for team members to communicate with each other as well as keep track of a schedule and informing them if any changes need to be made. (Emily)

Tester: The tester is responsible for in this case creating the mock databases that will need to be created to test the system. In addition, the tester will be required to write test code that will specifically look for problem areas in the system or ways to optimize the system. In addition, this person will solicit potential users to test the "ease of use" of the system and report the user feedback to the rest of the team. (Neelu)

Coders: The remainder of the team members will be coders who work specifically on one aspect of the system. There should be at least one person working on each of these areas: HTML/java scripting, server, client, database interactions. (Rikard, Lisa, Brian, David, Brett)

6.0 Schedule

The following is a tentaive schedule for implementation of the BEAR/Advisor:

3/6 Deadline for Team members to decde what role they will be playing in the development process. Team web site should be online by this date.

3/16 Solidification of exact nature of user interface. Decide on how exactly to resolve the problem of having to getting access to data

3/20 Deadline for low-level designs for individual parts of the program

3/30 All team members should have reviewed the low level design. Skeleton classes including interfaces between different parts of the program should be set up. System manager should be implemented so that team can start compiling program to each others directories.

4/5 Basic client/server interaction should be working. Prototype for UI implemented.

4/20 Trial integration: All high priority features should be fully functional (pre-registration, adding/dropping, etc.

4/27 Complete debugging high priority functionality, begin getting user feedback

5/1 Integration II: Mid/low level priority functionality

5/8 Complete debugging for mid/low functionality

5/12 Integration of changes in system from user comments completed. Tartget date for completed, functional system.

5/20 End of beta testing, project is complete.

7.0 Unresolved Issues in the Specifications Document

The specs for both Lisa and Melissa were extremely thorough. The only questions I really have, and this is probably because i know nothing about it, are in regard to security. I'm still not exactly sure how this is going to be implemented and what we will need to do it. Also I'm not sure what kind of database we are going to use and how much of that code we are going to have to write ourselves. But I get a really good sense of what the User Interface looks like and how it works as well as what the features of the system will be.